A Partition-Based Random Search Method for Multimodal Optimization

Ziwei Lin, Andrea Matta, Sichang Du, Evren Sahin

mathematics

MDPI

Article

A Partition-Based Random Search Method for Multimodal Optimization

Ziwei Lin [1], Andrea Matta [1], Sichang Du [2] and Evren Sahin [3,*]

[1] Politecnico di Milano, Department of Mechanical Engineering, 20133 Milan, Italy
[2] Department of Industrial Engineering and Management, School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
[3] Laboratoire Genie Industriel, CentraleSupelec, Paris Saclay University, 3 Rue Joliot-Curie, 91192 Gif-sur-Yvette, France
* Correspondence: evren.sahin@centralesupelec.fr

Abstract: Practical optimization problems are often too complex to be formulated exactly. Knowing multiple good alternatives can help decision-makers easily switch solutions when needed, such as when faced with unforeseen constraints. A multimodal optimization task aims to find multiple global optima as well as high-quality local optima of an optimization problem. Evolutionary algorithms with niching techniques are commonly used for such problems, where a rough estimate of the optima number is required to determine the population size. In this paper, a partition-based random search method is proposed, in which the entire feasible domain is partitioned into smaller and smaller subregions iteratively. Promising regions are partitioned faster than unpromising regions, thus, promising areas will be exploited earlier than unpromising areas. All promising areas are exploited in parallel, which allows multiple good solutions to be found in a single run. The proposed method does not require prior knowledge about the optima number and it is not sensitive to the distance parameter. By cooperating with local search to refine the obtained solutions, the proposed method demonstrates good performance in many benchmark functions with multiple global optima. In addition, in problems with numerous local optima, high-quality local optima are captured earlier than low-quality local optima.

Keywords: multimodal optimization; multiple optima; partition-based random search; niching

MSC: 90B40; 90-08

Citation: Lin, Z.; Matta, A.; Du, S.; Sahin, E. A Partition-Based Random Search Method for Multimodal Optimization. *Mathematics* **2023**, *11*, 17. https://doi.org/10.3390/math11010017

Academic Editor: Ioannis G. Tsoulos

Received: 3 November 2022
Revised: 7 December 2022
Accepted: 9 December 2022
Published: 21 December 2022

1. Introduction

Most optimization algorithms can only provide one of the optimal solutions when it is applied, even if more than one optimal solution may exist. Nevertheless, in some situations, finding multiple optimal solutions is desired, for example the following:

- The global optimal solution is not always implementable in real-world problems, due to some unforeseen physical, financial, or political constraints, such as the availability of some critical resources in the future and the dynamic environment in path-planning problems. Knowing multiple good alternatives is helpful for decision-makers to switch the solutions quickly when needed. For instance, if a machine fails during production and its repair is time-consuming, the decision-maker can quickly change the production plans without using the machine during that time.
- It is common in practice that some issues are difficult to formulate into the objective function or to evaluate exactly, such as the sensitivity of the selected machine parameters, the maintenance policy, and the preferences of decision-makers. In this situation, having a set of different and good alternatives in advance is often desired for further analysis.

- In some cases, it is critical to determine all highly valued areas (or all possibilities), such as the contaminant source identification in the water distribution network [1].
- In surrogate-based optimization methods [2,3], the promising solution pointed out by optimizing the constructed surrogate model is simulated iteratively. Finding multiple promising solutions in one iteration allows the methods to take advantage of parallel computing to run several time-consuming simulations simultaneously.
- Finding different locations of the peaks of the objective function in the search space can indicate some structural knowledge of the optimized function and provide some helpful insights into the properties of the studied system.

Multimodal optimization problem is concerned with locating multiple optima in one single run [4]. The aim of this paper is to deal with multimodal optimization problems, seeking multiple global optimal solutions and high-quality local optimal solutions (local optimal solutions with good objective function values) of the given problem. The benefits of applying multimodal optimization have been studied in many fields [5], such as seismological problems [6], metabolic network modeling [7], job-shop scheduling [8], virtual camera configuration problems [9], and feature selection [10].

In order to handle multimodal optimization tasks, classic optimization methods can be applied in multiple runs hoping to find different optima. Nevertheless, the same optimal solution may be obtained in different runs. If all m optimal solutions have the same probability to be found, the expectation of the number of optimization runs required to locate all optima is $m^{(m-1)}/(m-1)!$. This value is usually much larger in practice since one of the optima may have higher probability to be discovered than others. To avoid converging to the same solution, a common approach is that if one optimal solution is determined, the fitness (i.e., the objective function values) in the observed region is depressed in subsequent runs so that different optimal solutions can be found sequentially, e.g., [11,12]. Still, at least the same number of optimization runs as the number of the optimal solutions are required. In addition, if the fitness derating function and the distance parameter that defines the neighbor range are not carefully selected, it may result in elimination of other optima that have not been found, or spurious optima caused by the modified objective function, although the occurrence of spurious optima can be reduced by incorporating a local search method based on the original objective function, e.g., sequential niching memetic algorithm (SNMA) [13]. Approaches without the determination of the neighbor radius can also be found in the literature, but a lot of effort is spent in an additional sampling of interior points, e.g., [14,15].

Evolutionary algorithms (EAs), e.g., genetic algorithm (GA) [16], particle swarm optimization (PSO) [17], and differential evolution (DE) [18], have the ability to preserve multiple solutions during the optimization process. With the help of niching techniques, they are capable of capturing multiple optima in a single run. Niching techniques were originally developed to preserve the population diversity and reduce the impact of the genetic shift. They are also used in multiobjective optimization problems to search for the Pareto-optimal set, e.g., nondominated sorting GA II (NSGA-II) [19]. In multimodal optimization problems, the use of niching techniques promotes population diversity, allowing multiple optima to be found and maintained. Among the niching techniques in the literature, the clearing procedure [20] eliminates neighbors before the selection until only a few dominating individuals remain in the clearing radius. Singh and Deb [21] reallocate the cleared individuals outside the clearing radius, hoping to find other areas of interest. Fitness sharing methods [22–24] depress the fitness of densely located individuals according to the population density within the sharing radius. Clustering algorithms, e.g., k-means method [25], can be used in fitness sharing methods for the formation of niches to reduce the computational cost and avoid the determination of sharing radius, e.g., [26]. In crowding approaches [27,28], the new generated individual replaces the most similar individual to maintain the initial diversity, if better fitness is observed. In restricted tournament selection [29], the new generated individuals compete with the nearest individuals in a subpopulation randomly sampled from the population. In species conservation [30,31],

the species seeds dominating other individuals in the same species are conserved into the next generation and updated iteratively. Li [32] designed a ring topology [33] on the particle swarm and used the *lbest* PSO to create small niches without niching parameters. A detailed survey on some basic niching techniques in multimodal optimization can be found in [34]. In addition, several niching mutation operator strategies for DE have been proposed for multimodal optimization problems recently. For example, local-binary-pattern-based adaptive DE (LBPADE) [35] uses the local binary pattern operator to identify multiple regions of interests; distributed individuals DE (DIDE) [36] constructs a virtual population for each individual so that each individual can search for its own optima; automatic niching DE (ANDE) [37] locates multiple peaks based on the affinity propagation clustering.

In the aforementioned niching techniques, the entire population evolves together and genetic operators are designed to preserve the population diversity. In contrast, some niching algorithms divide the entire population into parallel subpopulations, including forking GA [38], multinational GA [39], multipopulation GA [40], NichePSO [41], speciation-based PSO (SPSO) [42], swarms [43,44], culture algorithm with fuzzing cluster [45], LAM-ACO [46], and dual-strategy DE (DSDE) [47]. Each subpopulation evolves independently, searching for its own optimum. Different from classic EAs with multiple runs, subpopulations will be merged, split, interchanged, or reformed during the search process according to the positions of the individuals in the entire population. As a consequence, repeated convergence and inefficiency search are avoided.

Most existing niching techniques are sensitive to the selected parameters, which are usually application-dependent and may be heterogeneous in the search space, such as the radius parameters in clearing and fitness sharing, the species distance in species conservation, the window size in restricted tournament selection, and the number of seeds in clustering algorithms. Adaptive methods are studied to make the algorithms more robust to the distance parameters or to let the parameters vary in the search space, e.g., [48–50]. Some approaches are developed to detect whether two individuals belong to the same peak without the use of distance parameters, such as the hill–valley function [39] and the recursive middling [51]. Nevertheless, a great amount of additional fitness evaluations are required, significantly reducing the efficiency of the algorithm. The formation of the niches is still a challenge in the multimodal optimization community.

Recently, some researchers solved the multimodal optimization problem by converting it into a multiobjective optimization problem, named *multiobjectivization* [52]. For instance, biobjective multipopulation genetic algorithm (BMPGA) [53,54] uses the average absolute value of the gradient as another ranking criterion, in addition to the original objective function, for elitism and selection in the GA framework, thus providing a chance for survival for local optima. Deb and Saha [55,56] created a second objective function (e.g., the norm of the gradient or the number of better neighboring solutions) so that all optima are located in the weak Pareto-optimal set. Then, the modified NSGA-II algorithm [19], developed for multiobjective optimization problems, was applied to find all the optima in a single run. Diversity indicators, such as the distance from other individuals and the density of niches, are also considered as the second objective function to maintain the population diversity (similar to the niching techniques), e.g., [57,58]. Conflicting objective functions were also designed to increase the efficiency of the applied multiobjective optimization algorithm, e.g., multiobjective optimization for multimodal optimization (MOMMOP) [59] and triobjective differential evolution for multimodal optimization (TriDEMO) [60].

All the multimodal optimization methods mentioned above are developed under the framework of EAs, in which the population size should be determined based on the number of desired optima. However, the number of optima is usually unknown before executing the algorithm, although in some cases it can be estimated from prior knowledge about the system. Saving the obtained optima in an archive and reinitializing the individuals can extend the optimal solution set, but it may cause the population to converge to the previous found solutions.

Moreover, when dealing with the local optima, the existing multimodal optimization methods may fall into the following situations:

- Can only find multiple global optimal solutions (e.g., MOMMOP [59] and TriDEMO [60]), i.e., the local optimal solutions cannot be found.
- Assume that the global optimal solutions and local optimal solutions with different objective function values have the same importance (e.g., [55,56]).
- Prefer local optimal solutions in sparse areas to local optimal solutions with good objective function values (e.g., EAs with niching).
- Prior knowledge to determine the threshold (or tolerance) of the objective function value to decide whether to save a solution or not.

Therefore, when the computational time is limited, these methods cannot meet the demand of searching only the global optimal solutions and high-quality local optimal solutions (i.e., local optimal solutions with good objective function values) without prior knowledge.

A completely different approach, which requires no prior knowledge about number of optima in the studied problem, is proposed in this paper. In the proposed method, the feasible domain is partitioned into smaller and smaller subregions iteratively. At each iteration, solutions from different subregions are sampled and evaluated. Based on the observed information, different regions are partitioned at different rates. By controlling the partition rates of different regions, promising areas are exploited and reach the smallest size (e.g., singleton regions in discrete cases or regions with acceptable precision in continuous cases) earlier than nonpromising areas. Multiple promising areas can be partitioned in parallel, allowing multiple optimal solutions to be found in a single run. If the available budget size (the budget size indicates the number of evaluations of the objective function) is unlimited, all areas of the feasible domain will eventually be partitioned into subregions of the smallest size, i.e., all optima (both global and local) will be discovered eventually.

Partition-based random search methods, such as nested partition (NP) [61], COMPASS [62] and adaptive hyperbox algorithm [63], are efficient in optimization problems with large search space, i.e., the feasible range of the decision variable is large compared to its desired precision. The entire domain is partitioned into subregions, based on previous observations, trying to guide the search towards a promising region. However, in most partition-based methods, only the most promising region can be identified and stored; thus, only one optimal solution can be found. The probabilistic branch and bound (PBnB) [64,65] is developed to locate a subset of feasible solutions whose objective function values reach the given quantile level. Different from our approach, the partition rates are homogeneous in the search space in the PBnB. All regions are partitioned at the same rate until they are pruned (statistically, no solutions in this region belong to the subset of interest) or maintained (statistically, all solutions in this region belong to the desired subset). The PBnB method may also be extended to find multiple global optimal solutions by setting an extremely small quantile level. However, this will result in a large budget spent on estimating quantiles.

The classification of related works and the difference compared to the proposed algorithm are summarized in Table 1. A previous version of the proposed algorithm was presented in a conference paper [66], which mainly focused on searching for the global optimal solution under the interference of a large number of local optimal solutions. The algorithm is extended in this paper to find and store multiple global optimal solutions as well as high-quality local optimal solutions, including a scheme to extract the optimal solutions and a local search to refine the extracted optimal solutions during the optimization process. The research contributions of this paper are summarized below:

- Estimating the number of optima is not needed before performing the proposed method (which is required in all EA-based multimodal optimization methods). All optimal solutions (both global optimal solutions and local optimal solutions) will be discovered subsequently as the algorithm proceeds.

- Given a computational time, global optimal solutions and high-quality local optimal solutions are captured with higher probabilities than low-quality local optimal solutions, and no prior knowledge about the objective function is needed. Current multimodal optimization methods either cannot find local optimal solutions or treat all optimal solutions as equally important.
- The proposed method can also handle the cases in which the optimal solutions are regions rather than single points, i.e., there exists a region in the feasible domain where all solutions are optimal. The density of the solutions in the region depends on the user-defined precision level. This is new compared to all multimodal optimization methods.

The proposed method is tested in benchmark functions. The numerical results show that the proposed method works as expected and demonstrates good performance compared to other state-of-the-art multimodal optimization methods in the literature.

Table 1. Classification of selected related works.

Related Works	Algorithm Framework	Multimodal Optimization	Single Run
[12]	Sequential runs with tricks	✓	
[14]	Sequential runs with tricks	✓	
[15]	Sequential runs with tricks	✓	
LBPADE [35]	EA with niching	✓	✓
DIDE [36]	EA with niching	✓	✓
ANDE [37]	EA with niching	✓	✓
SPSO [42]	EA with subpopulations	✓	✓
LAM-ACO [46]	EA with subpopulations	✓	✓
DSDE [47]	EA with subpopulations	✓	✓
[55,56]	Multiobjectivization (EA)	✓	✓
BMPGA [53,54]	Multiobjectivization (EA)	✓	✓
MOMMOP [59]	Multiobjectivization (EA)	✓	✓
NP [61]	Partition-based random search		✓
PBnB [65]	Partition-based random search		✓
The proposed method	Partition-based random search	✓	✓

This paper is organized as follows. Section 2 describes the proposed method in detail. Section 3 combines the proposed method with a local search method to improve the efficiency of the algorithm. Numerical results on benchmark functions are discussed in Section 4. An engineering case showing the application of the proposed method is presented in Section 5. Finally, conclusions and future developments are presented in Section 6.

2. Proposed Method

Without loss of generality, a minimization problem is considered: $\min_{\boldsymbol{x}} f(\boldsymbol{x})$. The objective function is deterministic and the feasible domain is denoted as $\mathcal{D}$, where $\mathcal{D} \subset \mathbb{R}^d$.

In the proposed method, the entire feasible domain $\mathcal{D}$ is iteratively partitioned into subregions $\mathcal{D}_k$ such that $\cup_k \mathcal{D}_k = \mathcal{D}$ and $\cap_k \mathcal{D}_k = \emptyset$. Each subregion contains a set of feasible solutions. In most partition-based optimization methods, the extreme value, the mean, or the quantile of the sampled values, i.e., the objective function values of sampled solutions, are commonly used to rank the regions. Compared to the extreme value, the quantile contains more global information about the region so that the influence of outliers can be mitigated. Compared to the mean, the quantile focuses more on the good part of solutions in this region, rather than the overall region. Therefore, the quantile is adopted as the ranking criterion in this paper. The lower the estimated α-quantile, where $\alpha < 0.5$, the more promising the region. A promising region is further partitioned into smaller subregions so that this region can be further exploited. Multiple promising regions can be partitioned in parallel in order to obtain a set of optimal solutions. More specifically, different partition rates are adopted for different regions, and promising regions would be partitioned faster (exploited earlier) than nonpromising regions.

In this paper, we define the partition rate of a region as the reciprocal of the number of iterations required for this region to be further partitioned. The idea of controlling the partition rates for different regions is realized with the help of a budget allocation strategy [67], which is introduced for the sake of completeness in Section 2.1. The computational complexity of the proposed method is analyzed in Section 2.3. Section 2.2 describes the proposed method in detail and a simple example is introduced in Section 2.4 to give the reader a better understanding of the proposed method.

2.1. Budget Allocation for Quantile Minimization (BAQM)

The BAQM method [67] is proposed to allocate a budget of size N to K groups, i.e., sample N values from K groups, aiming to minimize the α-quantile of all the sampled values. The budget is allocated to each group dynamically, based on the previous observations, trying to let the sample size in group k be approximately proportional to its posterior probability of being the best group, i.e., the group having the lowest α-quantile. This budget allocation method is developed on the assumption that the values sampled from a group are independently, identically, and normally distributed. Nevertheless, the numerical results show that it also has good performance, even if the normality assumption is not satisfied.

Assume that for any group k, $n_{1,k}$ values have been observed and the group sample mean $\hat{\mu}_k$ and the group sample variance $\hat{\sigma}_k^2$ are calculated based on the $n_{1,k}$ observations. According to [67], at each sampling stage, a new budget of size Δ should be allocated using the following equations:

$$\frac{n_k}{n_{\hat{b}}} = \frac{F(C_{k,\hat{b}}; n_{1,k}-1, n_{1,\hat{b}}-1)}{F(C_{\hat{b},k}; n_{1,\hat{b}}-1, n_{1,k}-1)}, \forall k \neq \hat{b}, \tag{1}$$

$$n_{\hat{b}} = \left(\Delta + \sum_{\forall k} n_{1,k}\right) \Big/ \left(\sum_{\forall k} \frac{F(C_{k,\hat{b}}; n_{1,k}-1, n_{1,\hat{b}}-1)}{F(C_{\hat{b},k}; n_{1,\hat{b}}-1, n_{1,k}-1)}\right), \tag{2}$$

where n_k denotes the total budget size allocated to group k, $\hat{b}$ is the current best group defined as $\hat{b} = \arg\min_k\{\hat{\mu}_k + z_\alpha \hat{\sigma}_k\}$, z_α is the α-quantile of the standard normal distribution, $\hat{\tau} = \min_k\{\hat{\mu}_k + z_\alpha \hat{\sigma}_k\}$, $F(\cdot; v_1, v_2)$ is the cumulative distribution function of the F-distribution with degrees of freedom v_1 and v_2,

$$C_{i,j} = \frac{1 + \frac{1}{\hat{c}_j^2} - \frac{1}{n_{1,j}}}{1 + \frac{1}{\hat{c}_i^2} - \frac{1}{n_{1,i}}}, \forall i, j \tag{3}$$

and $\hat{c}_k = \frac{\hat{\sigma}_k}{\hat{\mu}_k - \hat{\tau}}, \forall k$. Then, additional $\max(0, [n_k] - n_{1,k})$ values should be sampled from group k at this sampling stage, where $[\cdot]$ indicates that the value is rounded to the nearest integer.

The BAQM method is easy to implement and it links the sample size to the ranking of groups defined by quantiles. Therefore, it is selected as the budget allocation strategy in the proposed method.

2.2. Partition-Based Random Search for Multimodal Optimization

Denote by $\mathbb{D}^s = \{\mathcal{D}_i^s\}$ the set of stored regions in iteration s. By regarding a region $\mathcal{D}_i^s$ as a group, the BAQM method can be applied to sample solutions from different regions dynamically. Gradually, the sample sizes in different regions, denoted by $n_i^s, \forall i$, will be approximately proportional to their posterior probability of being the best region, i.e., the region having the minimal α-quantile, based on the previous observations. If we set a threshold $n_{\max}$, the sample sizes in promising regions will achieve this threshold faster than that in nonpromising regions because more samples are allocated. Therefore, we further

partition a region when its sample size reaches the threshold. This makes the promising regions have a higher partition rate than the nonpromising regions.

In the proposed method, the data observed in previous iterations are reused. This allows the deviations caused by the sampling noise to be transmitted to subsequent iterations. To mitigate the influence of the sampling noise, the BAQM formulas are modified. Denote by l_i^s the partition depth of a region $\mathcal{D}_i^s$. In this paper, the partition depth of a region refers to the minimum number of partition actions required to obtain this region, which is often related to the size of the region. The smaller the region size, the larger the partition depth. The adjusted sample size n_i^{adj} is calculated based on the partition depth as follows:

$$n_i^{\text{adj}} = \max\left(2, \left[\frac{l_i^s}{\max_j\{l_j^s\}} n_i^s\right]\right), \forall i, \tag{4}$$

where n_i^s is the number of observations in region $\mathcal{D}_i^s$ and $[\cdot]$ indicates that the value is rounded to the nearest integer. This modification aims at forcing the method to also sample from nonpromising but broad regions. As all regions shrink, i.e., all l_i^s increase, and the influence of Equation (4) will decrease. Then, according to the BAQM formulas, the weight assigned to region $\mathcal{D}_i^s$ is calculated as follows:

$$w_i = \frac{F(C_{i,\hat{b}}; n_i^{\text{adj}} - 1, n_{\hat{b}}^{\text{adj}} - 1)}{F(C_{\hat{b},i}; n_{\hat{b}}^{\text{adj}} - 1, n_i^{\text{adj}} - 1)} = \frac{1}{1 - F(C_{i,\hat{b}}; n_i^{\text{adj}} - 1, n_{\hat{b}}^{\text{adj}} - 1)} - 1, \forall i, \tag{5}$$

where

$$C_{i,\hat{b}} = \frac{1 + z_\alpha^2 - 1/n_{\hat{b}}^{\text{adj}}}{1 + \left(\frac{\hat{\mu}_i - \hat{\tau}}{\hat{\sigma}_i}\right)^2 - 1/n_i^{\text{adj}}}, \forall i, \tag{6}$$

$\hat{b}$ is the current best region defined as $\hat{b} = \arg\min_k\{\hat{\mu}_k + z_\alpha \hat{\sigma}_k\}$, z_α is the α-quantile of the standard normal distribution, and $\hat{\mu}_i$ and $\hat{\sigma}_i^2$ are the sample mean and sample variance of the objective function values of solutions sampled from region $\mathcal{D}_i^s$, respectively. $\hat{\tau} = \hat{\mu}_{\hat{b}} + z_\alpha \hat{\sigma}_{\hat{b}}$, $F(\cdot; v_1, v_2)$ is the cumulative distribution function of the F-distribution with degrees of freedom v_1 and v_2. Given the new budget size Δ, the theoretical total budget sizes in each region n_i can be calculated as follows:

$$n_i = \left(\Delta + \sum_i n_i^{\text{adj}}\right) \cdot \frac{w_i}{\sum_i w_i} \tag{7}$$

The proposed method is very straightforward and easy to implement. The flow chart is as shown in Figure 1 and the main framework is presented in Algorithm 1. Four algorithm parameters are required:

α	The quantile level in the definition of the best region, $0 < \alpha < 0.5$;
n_0	The base sample size, $n_0 \geq 2$;
$n_{\max}$	The sample size threshold to further partition a region, $n_{\max} > n_0$;
Δ	The new budget size at each iteration.

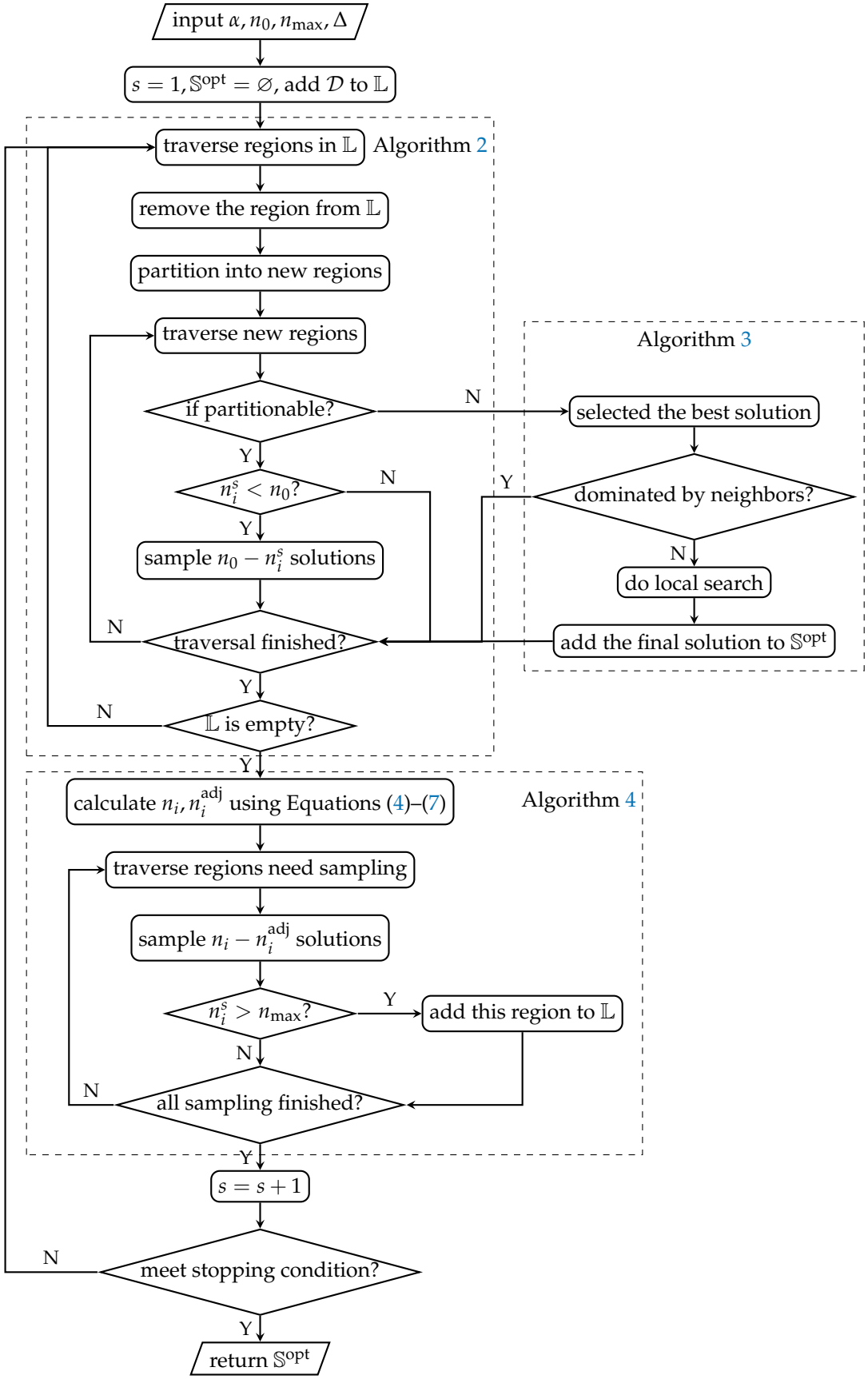


Figure 1. The flow chart of the proposed algorithm (Algorithm 1).

Algorithm 1 PAR- MMO.

Input: $\alpha, n_0, n_{\max}, \Delta, \mathcal{P}(\cdot), \mathcal{S}(\cdot,\cdot), U(\cdot), C^{\text{stop}}$
Output: $\mathbb{X}, \mathbb{S}^{\text{opt}}$

$s \leftarrow 1$
$\mathbb{D}^s \leftarrow \{\mathcal{D}_1^s = \mathcal{D}\}, n_1^s \leftarrow 0$
$\mathbb{L} \leftarrow \{1\}$
while $\neg C^{\text{stop}}$ **do**
  Partitioning (Algorithm 2)
  if $I^{\text{new}} = 1$ **then**
    Updating $\mathbb{S}^{\text{opt}}$ (Algorithm 3)
    $I^{\text{new}} \leftarrow 0$
  end if
  Budget Allocation (Algorithm 4)
  $s \leftarrow s + 1$
end while

Algorithm 2 Partitioning.

for $i \in \mathbb{L}$ **do**
  $\mathbb{D}^{\text{new}} \leftarrow \mathcal{P}(\mathcal{D}_i^s)$
  $\mathbb{D}^s \leftarrow \mathbb{D}^s \setminus \{\mathcal{D}_i^s\} \cup \mathbb{D}^{\text{new}}$
  for $j : \mathcal{D}_j^s \in \mathbb{D}^{\text{new}}$ **do**
    update n_j^s, l_j^s
    if $n_j^s \geq n_{\max}$ & $\neg I_{\mathbb{D}^*}(\mathcal{D}_j^s)$ **then**
      $\mathbb{L} \leftarrow \mathbb{L} \cup \{j\}$
    else
      if $n_j^s < n_0$ **then**
        $\mathbb{X} \leftarrow \mathbb{X} \cup \mathcal{S}(n_0 - n_j^s, \mathcal{D}_j^s)$
      end if
      update $\hat{\mu}_j, \hat{\sigma}_j^2$
      if $\neg I_{\mathbb{D}^*}(\mathcal{D}_j^s)$ **then**
        $n_j^{\text{adj}} \leftarrow$ Equation (4)
        $w_j \leftarrow$ Equation (5)
      else
        $n_j^{\text{adj}} \leftarrow 0$
        $w_j \leftarrow 0$
        $\mathbb{X}^{\text{opt}} \leftarrow \mathbb{X}^{\text{opt}} \cup \{\boldsymbol{x}_k : \boldsymbol{x}_k \in \mathcal{D}_j^s\}$
        $I_k^{\text{new}} \leftarrow 1$
      end if
    end if
  end for
end for
$\mathbb{L} \leftarrow \emptyset$

Algorithm 3 Extracting.

$i \leftarrow \min\{j : I_j^{\text{opt}} = 1\}$
while $i \neq \emptyset$ **do**
  $I_j^{\text{opt}} \leftarrow 0, \forall j \in \{k : \boldsymbol{x}_k \in U(\boldsymbol{x}_i) \cap \mathbb{X}^{\text{opt}}; f(\boldsymbol{x}_k) > f(\boldsymbol{x}_i)\}$
  if $f(\boldsymbol{x}_i) > \min_{j \in \{k:\boldsymbol{x}_k \in U(\boldsymbol{x}_i) \cap \mathbb{X}^{\text{opt}}\}} f(\boldsymbol{x}_j)$ **then**
    $I_i^{\text{opt}} \leftarrow 0$
  end if
  $i \leftarrow \min\{j : j > i; I_j^{\text{opt}} = 1\}$
end while
$\mathbb{S}^{\text{opt}} \leftarrow \{\boldsymbol{x}_i : I_i^{\text{opt}} = 1\}$

Algorithm 4 Budget allocation.

$N^{\text{adj}} \leftarrow \Delta + \sum_i n_i^{\text{adj}}$
$w = \sum_i w_i$
$n_i \leftarrow N^{\text{adj}} \cdot w_i / w, \forall i$
for $i : [n_i - n_i^{\text{adj}}] > 0$ **do**
  $\mathbb{X} \leftarrow \mathbb{X} \cup \mathcal{S}([n_i - n_i^{\text{adj}}], \mathcal{D}_i^s)$
  update n_i^s
  if $n_i^s \geq n_{\max}$ **then**
    $\mathbb{L} \leftarrow \mathbb{L} \cup \{i\}$
  else
    update $\hat{\mu}_i, \hat{\sigma}_i^2$
    $n_i^{\text{adj}} \leftarrow$ Equation (4)
    $w_i \leftarrow$ Equation (5)
  end if
end for

$\mathcal{P}(\text{region})$, $\mathcal{S}(\text{sample size}, \text{region})$, $U(\text{solution})$, and C^{stop} are the user-defined partition strategy, sampling strategy, neighborhood of the selected solution, and stopping criteria, respectively. The output $\mathbb{X}$ is the set of all sampled solutions and $\mathbb{S}^{\text{opt}}$ is the set of obtained optimal solutions. At the beginning of the algorithm, the set of the current regions $\mathbb{D}^s$ and the partition list $\mathbb{L}$ is set as the entire feasible domain $\mathcal{D}$. In the subsequent iterations, partitioning all regions in the partition list using Algorithm 2 and allocating budget to each region using Algorithm 4 are performed alternatively until the stopping criterion is met, such as the available budget is exhausted, the desired number of optimal solutions are obtained, or the number of obtained optimal solutions are not changed in several iterations. After the partitioning phase, if new potential optimal solutions are generated, i.e., $I^{\text{new}} = 1$, the set of optimal solutions $\mathbb{S}^{\text{opt}}$ are updated using Algorithm 3. This step can also be executed only at the end of the whole algorithm to save the computational effort, if $\mathbb{S}^{\text{opt}}$ is not related to the stopping criterion.

Algorithm 2 describes in detail the partitioning phase in Algorithm 1. $\mathbb{D}^*$ denotes the set of nonpartitionable regions and $I_{\mathbb{D}^*}(\cdot)$ is the indicator function. Every region in the partition list $\mathbb{L}$ is partitioned into several new subregions. The new subregions are added to the partition list if they are partitionable and their sample size n_j^s is still larger than or equal to the threshold $n_{\max}$. Otherwise, new solutions are sampled so that the sample size in this new subregion is not less than the base sample size n_0. Then, the group sample mean $\hat{\mu}_j$ and the group sample variance $\hat{\sigma}_j^2$ are updated. The adjusted sample size n_j^{adj} is calculated using Equation (4) and the weight for the budget allocation w_j is calculated using Equation (5). After traversing the entire partition list $\mathbb{L}$, $\mathbb{L}$ is set to an empty set.

Once a new nonpartitionable region is generated, the weight w_j and the adjusted sample size n_j^{adj} are set to zero, since the solutions within this region are considered not different; thus, it is not necessary to keep sampling from this region. If the current best region $\mathbb{D}_{\hat{b}}^s$ is nonpartitionable, $n_{\hat{b}}^{\text{adj}}$ is saved for the calculation of Equation (5). All the samples in the new generated nonpartitionable region are considered as potential optimal solutions; thus, they are added into the set of optima candidates $\mathbb{X}^{\text{opt}}$, and I^{new} is set to one to send the signal to run Algorithm 3.

Remark 1. *In Algorithm 2, if the maximal partition depth, i.e.,* $\max_i\{l_i^s\}$*, in Equation (4) is changed, the adjusted sample size* n_i^{adj} *of all regions should be recalculated. Thus, all weights* w_i *also need to be recalculated.*

Remark 2. *In Algorithm 2, if the current optimal region, i.e.,* $\hat{b}$*, in Equation (5) is changed, all weights* w_i *should be recalculated.*

Algorithm 3 presents the detailed procedure to extract the set of the optimal solutions $\mathbb{S}^{\text{opt}}$ from the set of potential optima $\mathbb{X}^{\text{opt}}$. In the algorithm, $I_i^{\text{opt}} = 1$ indicates that solution $\boldsymbol{x}_i$ is not dominated by the neighboring solutions. The I_i^{opt} values are set to one for all solutions newly added into $\mathbb{X}^{\text{opt}}$. Starting from the first solution with $I_i^{\text{opt}} = 1$, the I_j^{opt} values are set to zero for all the solutions $\boldsymbol{x}_j$ in the neighborhood whose objective function value is worse than the objective function value of the selected solution $\boldsymbol{x}_i$. If there exists a better solution in the neighborhood, the I_i^{opt} value of the selected solution is also set to zero. In optimization problems with continuous variables, a commonly used neighborhood function is $U(\boldsymbol{x}_i) = \{\boldsymbol{x} : ||\boldsymbol{x} - \boldsymbol{x}_i||_2 \leq r\} \cap \mathcal{D}$, where $||\boldsymbol{x} - \boldsymbol{x}_i||_2$ is the Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{x}_i$. In this case, Algorithm 3 is not sensitive to the selection of r.

Algorithm 4 describes in detail how to allocate a new budget of size Δ at each iteration. The adjusted total budget size N^{adj} is calculated by summing the adjusted region sample size n_i^{adj} and Δ. The theoretical total budget size at each region n_i is calculated using the weight w_i. Then, $\max(0, [n_i - n_i^{\text{adj}}])$ new solutions are sampled from region $\mathcal{D}_i^s$, where $[\cdot]$ indicates that the value is rounded into the nearest integer. Once the sample size in a region reaches the threshold $n_{\max}$, this region is added to the partition list $\mathbb{L}$.

2.3. Computational Complexity

As the total budget size increases, the number of existing regions grows as well, which will results in raised computational time in later iterations. The computational complexity of the proposed method is investigated in this section to understand the extent to which the total budget size affects the computational effort.

In Algorithm 2, the maximal length of the partition list is Δ in each iteration; thus, the time complexity of Algorithm 2 is $O(\Delta)$ in each iteration. The number of total iterations is less than N/Δ, where N is the total budget size allocated. Thus, the time complexity of Algorithm 2 is $O(N)$ in the entire optimization process.

Similar to Algorithm 2, the time complexity of lines 5–13 in Algorithm 4 is $O(\Delta)$ in each iteration and $O(N)$ in the entire optimization process, respectively. As for lines 1–4 in Algorithm 4, the n_i value should be calculated and compared to n_i^{adj} for all existing regions in each iteration. Thus, the time complexity is $O(|\mathbb{D}^s|)$ in iteration s. The number of existing regions $|\mathbb{D}^s|$ is less than $(h-1)\Delta s$, where h indicates the maximal number of subregions that will be generated after a partition action. Therefore, the time complexity of lines 1–4 becomes $O(\Delta s)$ in iteration s. In the entire optimization process, the time complexity is $O(\Delta \sum_{s=1}^{N/\Delta} s) = O(N + N^2/\Delta)$, i.e., it is increasing quadratically with respect to the total budget size N.

Algorithm 3 is only executed when new potential optimal solutions appear. Since the distance between every two solutions in the set of the potential optimal solutions $\mathbb{X}^{\text{opt}}$ should be calculated to determine the neighboring solutions, the time complexity of Algorithm 3 is $O(|\mathbb{X}^{\text{opt}}|^2)$ in the entire optimization process. Usually, the number of potential optima is much less than the total budget size.

Therefore, the overall time complexity of the proposed method with respect to the total budget size is $O(N^2/\Delta)$ due to line 3 and line 4 in Algorithm 4.

2.4. An Illustrative Example

Minimizing the Himmelblau's function is considered in this section as an illustrative example for the reader to better understand the proposed method. This problem has four global optimal solutions, and the objective function value varies from 0 to 2186 (the detailed information can be found in the selected benchmark function F2 in Appendix A Figure A1). The goal of this problem is to find and store all these four global optimal solutions.

The proposed method is applied with $n_0 = 4, \alpha = 0.3, n_{\max} = 10$, and $\Delta = 3$. The regions are evenly partitioned into half from the horizontal and vertical directions iteratively until all edges of all regions are smaller than 0.05, which are considered as nonpartitionable regions. Figure 2 presents the partition states on the entire feasible domain as the budget

size N increases. The sampled solutions are shown by dots, where their colors represent their objective function values. The darker the color, the lower the objective function value.

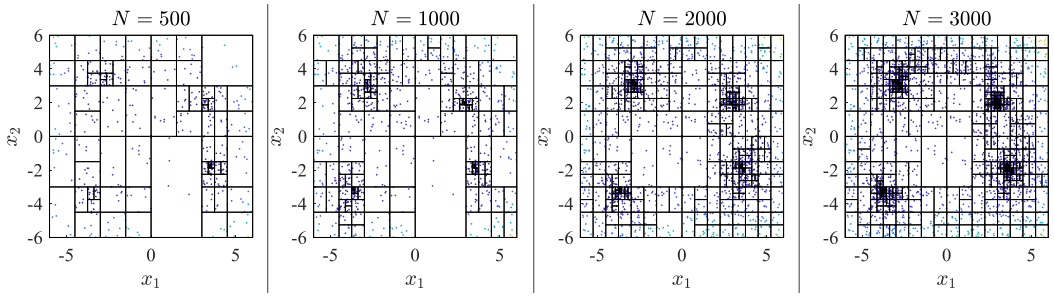


Figure 2. The partition states and the sampled solutions as the budget size N increases in the Himmelblau function minimization problem.

It can be seen that as the budget size increases, regions containing good solutions are partitioned at higher rates than other regions. The areas around the four global optimal solutions are exploited in parallel and reach the smallest size earlier than other areas. The sampling of solutions is guided by the proposed method. At the beginning of the algorithm, solutions are sampled evenly among the entire domain. As the budget size increases, the sampling probability of good solutions increases as well.

After about 3000 solutions are evaluated, Algorithm 3 is performed with $U(\boldsymbol{x}_i) = \{\boldsymbol{x} : ||\boldsymbol{x} - \boldsymbol{x}_i||_2 \leq r\} \cap \mathcal{D}$, where $r = 0.0938$, i.e., twice the length of the shortest edge of a nonpartitionable region. Four solutions are contained in $\mathbb{S}^{\text{opt}}$ and their objective function values are all less than 6×10^{-3}. The distances from the four solutions to their corresponding real optimal solutions are all less than 0.014. The same results can be obtained with r varying from 0.042 to 3.9, which shows that Algorithm 3 is not sensitive to the selection of the distance parameter r.

3. Cooperating with Local Search

The proposed partition-based random search method behaves conservatively. It can maintain a global perspective, thereby reducing the probability of losing some optimal solutions. It can be found in Section 2.4 that the proposed method detects the four promising areas fast, but it takes a lot of effort to obtain a precise optimal solution in the detected promising area. The search is always guided by the thought that there may be multiple local optima in a region. Thus, it cannot focus on exploitation to improve the accuracy of the obtained optimum in a detected promising area (also called detected peaks in maximization problems), especially when the promising area is relatively flat, i.e., the difference between the regions in the promising area is small.

In contrast, the local search method is efficient in locating the precise local optimal solution if the starting point is located nearby. The main issue of adopting local search in a multimodal optimization problem is the number and the locations of the starting points. If the starting points are not located carefully, it may result in loss of optima or waste of budget caused by multiple starting points within the same promising area, whereas the set $\mathbb{S}^{\text{opt}}$ extracted from Algorithm 3 contains different good solutions from different promising areas, which provides multiple good locations for the local search to start from. Therefore, the proposed partition-based random search can be used to detect promising areas, from which a local search is utilized to refine the solution in the set of obtained optimal solutions $\mathbb{S}^{\text{opt}}$ to obtain more precise optimal solutions. A similar idea appears in EMO-MMO [68], in which an algorithm is developed to detect peaks from solutions sampled by multiobjectivization methods and a swarm-based method is used within each peak.

Any new solution that appears in $\mathbb{S}^{\text{opt}}$ after executing Algorithm 3 is used as the starting point in a local search method to obtain a more precise optimal solution with higher accuracy. Algorithm 5 introduces a simple local search method for problems with

continuous domain. The current solution iteratively moves to the best neighboring solution with one step difference in one dimension. In the algorithm, $\boldsymbol{e}_j$ is a d-dimensional vector whose j-th element is one and the rest of the elements are all zero. The initial step δ is set as the distance parameter r in Algorithm 3 and it is shrinking as the search proceeds. The local search is repeated until the stopping criteria is met, such as when the step δ is smaller than a threshold δ^*, the desired objective function value is obtained, or the budget is exhausted. All the new sampled solutions are added into the set of optima candidates $\mathbb{X}^{\text{opt}}$ with $I_i^{\text{opt}} = 0$, except the local optima whose I_i^{opt} is assigned to one. Algorithm 5 is an example of how the accuracy of the obtained optima can be improved. Other optimization algorithms with strong local search capacity can also be applied according to the features of the studied problem.

Algorithm 5 Local search.

Input: $\boldsymbol{x}_i \in \mathbb{S}^{\text{opt}}, \delta^*$
Output: $\boldsymbol{x}_i^*$

$\boldsymbol{x}_i^* \leftarrow \boldsymbol{x}_i$
$\delta \leftarrow r$
$C_2^{\text{stop}} \leftarrow 0$
while $C_2^{\text{stop}} = 0$ **do**
  $\boldsymbol{x}' \leftarrow \boldsymbol{x}_i^*$
  for $j = 1, \cdots, d$ **do**
    $\boldsymbol{x}_i^* \leftarrow \arg\min_{\{\boldsymbol{x}_i^* - \delta \boldsymbol{e}_j, \boldsymbol{x}_i^*, \boldsymbol{x}_i^* + \delta \boldsymbol{e}_j\}} f(\boldsymbol{x})$
  end for
  if $\boldsymbol{x}' = \boldsymbol{x}_i^*$ **then**
    if $\delta < \delta^*$ **then**
      $C_2^{\text{stop}} \leftarrow 1$
    end if
    $\delta \leftarrow \delta / 2$
  end if
end while
$\mathbb{S}^{\text{opt}} \leftarrow \mathbb{S}^{\text{opt}} \setminus \{\boldsymbol{x}_i\} \cup \{\boldsymbol{x}_i^*\}$

4. Numerical Results

The proposed method is applied to minimization problems constructed by several benchmark functions with different properties extracted from the well-known test problems of the CEC'2013 competition for multimodal optimization [34,69]. Table 2 shows the selected objective function, the dimension of the decision variable, the number of global optima and local optima, the feasible domain, the objective function value range, and the maximum budget size `Max_Fes` in different test problems. F1–F3 are simple examples with multiple global optima. F4 is a volatile function with numerous local optima. The optimal solutions in F5 are located in different topographies in the feasible domain. F6 contains multiple global optimal solutions distributed in a grid. F7 is composited by different basic functions such that the properties of different functions are mixed. F1–F7 contain multiple global optima, while F8 and F9 have only one global optimum and multiple local optima of different qualities. In F10, there are several regions in which all points are local optimal solutions. The detailed information and the surface plots of the benchmark functions can be found in Appendix A.

Table 2. The parameters of the tested problems.

Func	Dim	No.g	No.l	Domain	Value Range	Radius	Max_Fes
F1	1	5	0	$[0,1]$	$[0,1]$	0.015	5×10^4
F2	2	4	0	$[-6,6]^2$	$[0,2185.8]$	0.1	5×10^4
F3	2	2	4	$[-1.9,1.9],[-1.1,1.1]$	$[-4.1265,23.4]$	0.1	5×10^4
F4(2D)	2	18	>700	$[-10,10]^2$	$[-186.7309,210.5]$	0.01	2×10^5
F4(3D)	3	81	>2000	$[-10,10]^3$	$[-2709.0935,3053.7]$	0.01	4×10^5
F5(2D)	2	36	0	$[0.25,10]^2$	$[0,2]$	0.03	2×10^5
F5(3D)	3	216	0	$[0.25,10]^3$	$[0,2]$	0.03	4×10^5
$F6_{(3,4)}$	2	12	0	$[0,1]^2$	$[2,38]$	0.01	2×10^5
$F6_{(3,3,3)}$	3	27	0	$[0,1]^3$	$[3,57]$	0.01	4×10^5
$F6_{(2,\cdots,2)}$	5	32	0	$[0,1]^5$	$[5,95]$	0.02	4×10^5
F7(1-2D)	2	6	>500	$[-5,5]^2$	$[0,2204.2]$	0.01	2×10^5
F7(2-2D)	2	8	>600	$[-5,5]^2$	$[0,2050.2]$	0.01	2×10^5
F7(3-2D)	2	6	>2000	$[-5,5]^2$	$[0,3701.5]$	0.01	2×10^5
F7(3-3D)	3	6	>2000	$[-5,5]^3$	$[0,4126.1]$	0.01	4×10^5
F8	1	1	4	$[0.02,1]$	$[0,1]$	0.01	5×10^5
F9(2D)	2	1	120	$[-5.12,5.12]^2$	$[0,80.7]$	0.1	3×10^4
F9(3D)	3	1	1330	$[-5.12,5.12]^3$	$[0,121.1]$	0.1	1×10^5
F10	2	1	4 [a]	$[-10,10]^2$	$[0.0097,0.9975]$	-	-

[a] The local optima are not single points.

In the following experiments, if the deviation from the objective function value of an obtained optimal solution to the objective function value of a real optimal solution is below ε (level of accuracy) and the distance between these two solutions is less than the radius in Table 2 (level of precision), this real optimal solution is considered as being found. For the proposed method, a solution is considered as an obtained optimal solution only when it is stored in $\mathbb{S}^{\text{opt}}$ (]the datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request).

For the sake of simplicity, the proposed method, denoted as "PAR-MMO", is applied with $\alpha = 0.3, n_0 = 4, n_{\max} = 10$, and $\Delta = 3$ for all the following experiments, unless specifically stated. If a budget is allocated to a region, new solutions are sampled from this region uniformly (the sampling strategy). If the partitioning condition is met, the region is partitioned evenly into two subregions from the dimension with the largest range (the partition strategy). The accuracy level of the obtained optima is controlled by the acceptable precision of the obtained solution, i.e., the size of the nonpartitionable region. Once a region reaches the smallest size, i.e., it is nonpartitionable, the proposed method will stop sampling from this region and the accuracy of the obtained optima in this region will not be further improved. The smaller the region that is considered nonpartitionable, the more precise the solution obtained, and the larger the budget required. In the following experiments, the size of a nonpartitionable region is defined specific to the problem in order to met the required accuracy level ε (as shown in Appendix B). In practice, the size of the nonpartitionable region can be defined according to the acceptable precision of the solution obtained. The neighborhood function in Algorithm 3 for extracting optimal solutions from the sample set is selected as $U(\boldsymbol{x}_i) = \{\boldsymbol{x} : ||\boldsymbol{x} - \boldsymbol{x}_i||_2 \leq r\} \cap \mathcal{D}$, where r is twice the length of the shortest edge of a nonpartitionable region.

In the case that local search is adopted, denoted as "PARL-MMO", the size of the nonpartitionable region can be much larger (as shown in Appendix B), since both the accuracy level of the obtained optima and the precision of the obtained solution can be improved through the local search. In the following experiments, the step threshold δ^* in Algorithm 5 for local search is selected as the half length of the edge of the nonpartitionable region defined when the PAR-MMO method is used without local search.

Section 4.1 presents how the selected parameters affect the proposed method. In Section 4.2, the proposed method is applied to some problems with multiple global optima

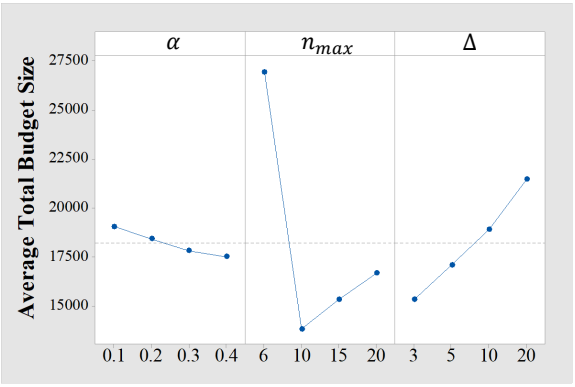
and compared with other approaches in the literature. Section 4.3 shows how the proposed method performs in the problems with multiple local optima of different qualities. Section 4.4 deals with problems that exist an region in which all solutions are optimal. The computational time of the proposed method is analyzed in Section 4.5.

4.1. Effect of Algorithm Parameters

This section investigates how the algorithm parameters, i.e., the quantile level α, the base sample size n_0, the partition sample size threshold $n_{\max}$, and the new budget size Δ, affect the proposed method. In this section, "PARL-MMO" is applied and the accuracy level ε is set as]1E-4. The base sample size n_0 is set to avoid the situation where too few samples remain in a subregion after a partition action. Thus, the n_0 is set as $\lceil n_{\max}/3 \rceil$, where $\lceil \cdot \rceil$ means that the value is rounded up to the nearest integer.

4.1.1. Main Effect Plot

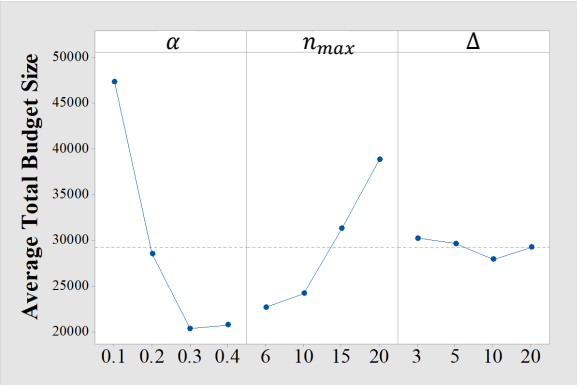
Figure 3 shows the main effect plot and the table of analysis of variance (ANOVA) after running a full factorial design in Problem $F6_{(3,3,3)}$ with three factors: $\alpha = \{0.1, 0.2, 0.3, 0.4\}$, $n_{\max} = \{6, 10, 15, 20\}, and \Delta = \{3, 5, 10, 20\}$. A total of 500 independent replications are executed and they are divided into 10 batches. The response is the average total budget size, which is required to obtain all global optima, in a batch. The ANOVA is implemented after the Box–Cox transformation so that the standardized residuals do not violate the normality assumption (the p-value equals 0.376 in the Anderson–Darling test) and the equal variance assumption (the p-value equals 0.983 in the Levene test). All the parameters and interactions have significant effect on the proposed method with significant level 5%. The influences of the $n_{\max}$ value and the Δ value are much larger than that of the α value and the interactions based on the F-values. Similar conclusions can be drawn in other problems (see Appendix C), except for Problem F5, where the optima are located in different topographies in the feasible domain, and Problem F7(2-2D). According to the Tukey pairwise comparison with confidence level 95%, the optimal combinations of parameters for Problem $F6_{(3,3,3)}$ are $\alpha = \{0.2, 0.3, 0.4\}$, $n_{\max} = 10$, and $\Delta = 3$.



Source	DF	F-Value	P-Value
α	3	66.37	0.000
n_{max}	3	5999.95	0.000
Δ	3	1674.54	0.000
$\alpha * n_{max}$	9	8.10	0.000
$\alpha * \Delta$	9	7.90	0.000
$n_{max} * \Delta$	9	30.38	0.000
$\alpha * n_{max} * \Delta$	27	2.08	0.002
Error	576		
Total	639		

Figure 3. The main effect plot and the ANOVA table. A total of 500 replications are executed and divided into 10 batches. The Box–Cox transformation is performed, and $R^2_{adj} = 98.67\%$.

Figure 4 shows the main effect plot and the ANOVA table for Problem F5(2D). Although the normality hypothesis and the equal variance hypothesis are not met with confidence level 95%, the F-test is robust since the experiments with all combinations are performed with the same number of replications. Different from Problem $F6_{(3,3,3)}$, the parameter α has the highest effect on the algorithm performance. According to the Tukey pairwise comparison with confidence level 95%, the optimal combinations of the algorithm parameters for Problem F5(2D) are $\alpha = 0.3$, $n_{\max} = 10$, and $\Delta = 3$.



Source	DF	F-Value	P-Value
α	3	11598.64	0.000
n_{max}	3	3677.19	0.000
Δ	3	47.75	0.000
$\alpha * n_{max}$	9	1719.85	0.000
$\alpha * \Delta$	9	605.19	0.000
$n_{max} * \Delta$	9	136.74	0.000
$\alpha * n_{max} * \Delta$	27	32.26	0.000
Error	576		
Total	639		

Figure 4. The main effect plot and the ANOVA table for Problem F5(2D). A total of 500 replications are executed and divided into 10 batches. The Box–Cox transformation is performed, and $R^2_{adj} = 99.08\%$.

4.1.2. Effect of the Partition Sample Size Threshold $N_{\max}$

As the $n_{\max}$ value increases, the average total budget size required to obtain all optima declines first and then rises. This is because when the $n_{\max}$ value is too small, the partition action is executed based on biased information due to the small sample size. When the $n_{\max}$ value is too large, the budget that could be used to exploit subregions with good performance is wasted on exploring the current region. A similar phenomenon can be observed in different problems, as shown in Figure 5, in which the budget size in the figure is scaled according to the maximal average budget size in different problems, i.e., the values in the legend.

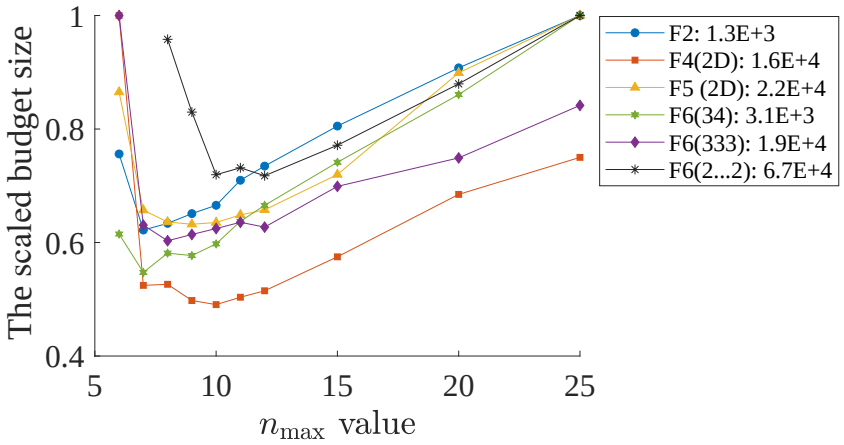


Figure 5. The scaled average budget size required to obtain all global optima in different problems with varying $n_{\max}$ value. A total of 100 replications are executed.

4.1.3. Effect of the New Budget Size Δ

The results of the ANOVA in Figure 3 show that a small Δ value is preferred in Problem $F6_{(3,3,3)}$ in terms of the total budget size, because it allows more budgets to be allocated after obtaining more information and generating more precise regions. Nevertheless, according to the discussion in Section 2.3, a small Δ value may increase the computational time of the proposed method. In Figure 6, the average total budget sizes and the corresponding computational times required to obtain all optima in Problem $F6_{(3,3,3)}$ with different Δ values are presented. The confidence intervals with confidence level 95% are also plotted. The experiments are executed in Matlab R2020a on a computer (Intel(R) Core(TM) i7-7700U CPU @ 3.6 GHz and 16 GB of RAM).

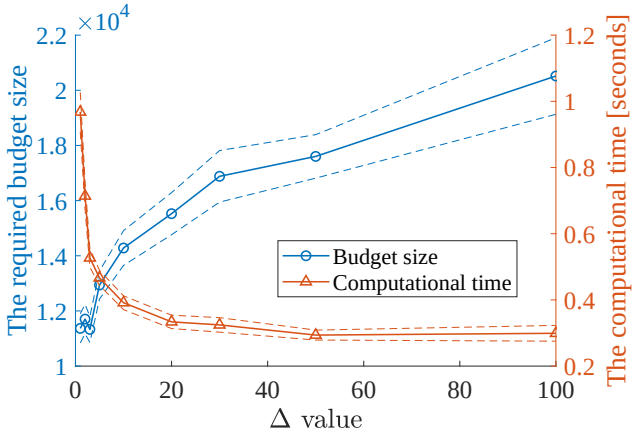


Figure 6. The average total budget size and the average computational time required to obtain all optima in Problem $F6_{(3,3,3)}$. A total of 100 replications are executed.

In this problem, it is very fast to calculate the objective function. Thus, the computational time is mainly affected by the computational complexity of the algorithm, i.e., $O(N^2/\Delta)$, where N is the total budget size. This is why the total computational time in Figure 6 decreases as the Δ value increases. In practice, if the calculation of the objective function is time-consuming, a small Δ value can be used to save the expensive budget, whereas if the calculation of the objective function is not critical, a relatively large Δ value can be used to reduce the computational time.

4.1.4. Effect of the Quantile Level α

The definition of promising regions is affected by the selected α value. A low α value prefers regions with high variability; thus, the proposed method will focus more on exploration to avoid the loss of some promising areas, whereas a high α value prefers regions with a low sample mean, so that the detected promising area will be firstly exploited. Although the effect of the α value is less significant than the other two parameters in Problem $F6_{(3,3,3)}$, it has a great influence in Problem F5(2D), where some optimal solutions are located in flat areas and others are located in steep areas. A low α value will make promising regions in flat areas struggle to reach the smallest size due to their small variability; thus, the samples in these regions are not considered as potential optimal solutions.

Figure 7 presents another influence of the α value in Problem F5(2D). In 100 replications, the frequencies of the optimal solutions in different locations being captured within a budget of size 8000 is represented by different colors and shapes. In the studied case, the size of nonpartitionable regions are the same among the whole domain. A promising region in flat areas (i.e., large x_1 and x_2 values) has a small sample mean, while a promising region of the same size in steep areas (i.e., small x_1 and x_2 values) has a large sample variance. When the α value is high ($\alpha = 0.3$), promising regions in flat areas will be partitioned earlier, and the global optimal solutions located in these areas can be found with a higher frequency (larger than 0.8). When the α value is reduced to a lower value (e.g., 0.1 or 0.2), the influence of the sample variance rises. Therefore, the frequency of finding the optimal solutions in flat areas decreases, whereas the frequency of finding the optimal solutions in steep areas increases.

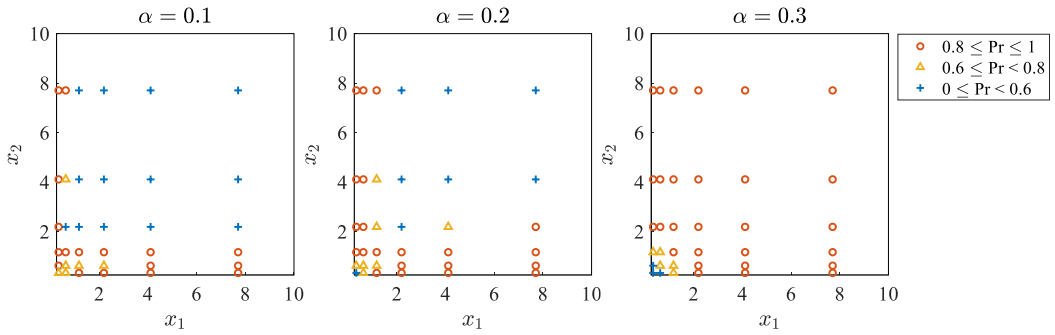


Figure 7. The frequencies of capturing different optimal solutions among 100 replications with varying α in Problem F5(2D). The budget size is 8000. As the α value increases, the frequency of finding optima in steep areas is reduced while the frequency of finding optima in flat areas is increased.

4.2. Comparison with Other Methods on Multiple Global Optima

The functions F1–F7, which have multiple global optima, are considered in this section. The proposed method is compared with other multimodal optimization methods developed from different mechanisms: multiobjectivization, subpopulations, differential evolution with niching mutation operator, and two-phase method. In addition, they all have good performance in the selected benchmark functions.

MOMMOP [59] transfers the multimodal optimization problem to a multiobjective optimization problem with $2d$ conflicting objective functions so that all optima are located in the Pareto front. Then, differential evolution combined with modified nondominated sorting [19] is used to solve the multiobjective optimization problem. The MOMMOP method belongs to the category of multiobjectivization and it is superior to ten state-of-the-art multimodal optimization algorithms in 20 benchmark functions.

LAMS-ACO [46] divides the entire population into subpopulations that evolve separately through a modified ant colony optimization algorithm ACO_R [70]. Random cluster size is adopted. A local search scheme is applied to refine the obtained solution at each iteration. The LAMS-ACO method belongs to the category of population-based niching using subpopulations and demonstrates good performance with respect to the required total number of evaluations compared to the other twelve multimodal optimization algorithms.

DIDE [36] constructs a virtual population for each individual so that each individual can track its own peak. The DIDE method belongs to the category of population-based method using niching mutation operator, and it has good performance compared to the other 13 methods in the benchmark functions.

EMO-MMO [68] develops an algorithm to detect the peaks from solutions sampled by a multiobjectivization method. Then, the competitive swarm optimizer (CSO) [71] is applied within each peak to refine the obtained optima. The idea of the EMO-MMO method is very similar to the proposed method; thus, it is also applied for comparison purposes.

These methods are applied with parameters suggested by the authors.

Three criteria are adopted for the assessment of the applied algorithms [69]. The first one is the peak ratio (PR), which measures the average percentage of the found global optima. The second one is the success rate (SR), which is the percentage of runs that find all the global optima. The third one is the convergence speed (CS), which is the average budget size, i.e., the average number of evaluations of the objective function, required to find all the global optima. If not all global optima are found when the budget is exhausted, the maximum budget size `Max_Fes` is used in the calculation.

The results of the applied algorithms are presented in Table 3 with the accuracy level ε varying from 1×10^{-1} to 1×10^{-4}. The winners are highlighted in bold and marked

with dark background color (determined through the Wilcoxon rank sum test with p-value smaller than 0.0125 for the PR and p-value smaller than 0.0167 for the CS). All the experiments are repeated 100 times independently. It should be noticed that the EMO-MMO method uses all the available budget; thus, the CS column is not presented.

Table 3. Comparisons (from top to bottom: $\varepsilon = 1 \times 10^{-1}$, 1×10^{-2}, 1×10^{-3}, and 1×10^{-4}).

	PAR-MMO			PARL-MMO			MOMMOP			LAMS-ACO			DIDE [1]		EMO-MMO	
Func	**PR**	**SR**	**CS**	**PR**	**SR**	**CS**	**PR**	**SR**	**CS**	**PR**	**SR**	**CS**	**PR**	**SR**	**PR**	**SR**
F1	1.00	1.00	1.6×10^2	1.00	1.00	$\mathbf{1.3} \times 10^2$	1.00	1.00	$\mathbf{1.4} \times 10^2$	1.00	1.00	$\mathbf{1.4} \times 10^2$	-	-	1.00	1.00
	1.00	1.00	2.8×10^2	1.00	1.00	$\mathbf{1.4} \times 10^2$	1.00	1.00	4.2×10^2	1.00	1.00	2.9×10^2	-	-	1.00	1.00
	1.00	1.00	3.7×10^2	1.00	1.00	$\mathbf{1.6} \times 10^2$	1.00	1.00	1.2×10^3	1.00	1.00	4.8×10^2	1.00	1.00	1.00	1.00
	1.00	1.00	5.5×10^2	1.00	1.00	$\mathbf{1.9} \times 10^2$	1.00	1.00	3.3×10^3	1.00	1.00	7.6×10^2	1.00	1.00	1.00	1.00
F2	1.00	1.00	2.4×10^3	1.00	1.00	$\mathbf{7.0} \times 10^2$	1.00	1.00	2.1×10^2	1.00	1.00	1.2×10^3	-	-	1.00	1.00
	1.00	1.00	3.7×10^3	1.00	1.00	$\mathbf{7.6} \times 10^2$	1.00	1.00	3.2×10^2	1.00	1.00	2.0×10^3	-	-	1.00	1.00
	1.00	1.00	7.1×10^3	1.00	1.00	$\mathbf{8.0} \times 10^2$	1.00	0.99	3.5×10^2	1.00	1.00	2.9×10^3	1.00	1.00	1.00	1.00
	1.00	1.00	1.2×10^2	1.00	1.00	$\mathbf{8.5} \times 10^2$	0.99	0.97	3.7×10^2	1.00	1.00	4.0×10^3	1.00	1.00	1.00	1.00
F3	1.00	1.00	3.4×10^2	1.00	1.00	$\mathbf{2.0} \times 10^2$	1.00	1.00	1.2×10^3	1.00	1.00	3.6×10^2	-	-	1.00	1.00
	1.00	1.00	7.5×10^2	1.00	1.00	$\mathbf{2.4} \times 10^2$	1.00	1.00	9.8×10^3	1.00	1.00	7.7×10^2	-	-	1.00	1.00
	1.00	1.00	1.2×10^3	1.00	1.00	$\mathbf{2.5} \times 10^2$	1.00	1.00	1.5×10^2	1.00	1.00	1.4×10^3	1.00	1.00	1.00	1.00
	1.00	1.00	2.1×10^3	1.00	1.00	$\mathbf{2.9} \times 10^2$	1.00	1.00	1.8×10^2	1.00	1.00	1.9×10^3	1.00	1.00	1.00	1.00
F4(2D)	1.00	1.00	2.4×10^2	1.00	1.00	$\mathbf{6.6} \times 10^3$	1.00	0.99	5.1×10^2	0.99	0.74	1.0×10^5	-	-	1.00	1.00
	1.00	1.00	3.4×10^2	1.00	1.00	$\mathbf{7.1} \times 10^3$	0.97	0.95	5.9×10^2	0.98	0.67	1.3×10^5	-	-	1.00	1.00
	1.00	1.00	5.5×10^2	1.00	1.00	$\mathbf{7.6} \times 10^3$	0.96	0.95	6.2×10^2	0.98	0.71	1.3×10^5	1.00	1.00	1.00	1.00
	1.00	0.97	9.2×10^2	1.00	1.00	$\mathbf{8.1} \times 10^3$	0.97	0.94	6.5×10^2	0.98	0.65	1.4×10^5	1.00	1.00	1.00	1.00
F4(3D)	0.55	0.00	4.0×10^5	**1.00**	0.98	2.1×10^5	**0.98**	0.94	2.3×10^5	0.77	0.00	4.0×10^5	-	-	**1.00**	**1.00**
	0.42	0.00	4.0×10^5	**1.00**	0.94	2.4×10^5	**0.99**	0.97	2.5×10^5	0.75	0.00	4.0×10^5	-	-	**1.00**	**1.00**
	0.26	0.00	4.0×10^5	**1.00**	0.97	2.2×10^5	**0.98**	0.95	2.8×10^5	0.73	0.00	4.0×10^5	0.69	0.00	**1.00**	**1.00**
	0.17	0.00	4.0×10^5	**1.00**	0.96	2.4×10^5	**0.98**	0.95	3.1×10^5	0.71	0.00	4.0×10^5	0.69	0.00	**1.00**	**0.99**
F5(2D)	1.00	0.97	3.7×10^2	1.00	1.00	$\mathbf{1.3} \times 10^2$	1.00	1.00	4.7×10^2	0.79	0.00	2.0×10^5	-	-	1.00	0.98
	1.00	1.00	4.9×10^2	1.00	1.00	$\mathbf{1.3} \times 10^2$	1.00	1.00	5.5×10^2	0.76	0.00	2.0×10^5	-	-	1.00	0.98
	1.00	0.89	1.1×10^5	1.00	1.00	$\mathbf{1.4} \times 10^2$	1.00	1.00	6.4×10^2	0.69	0.00	2.0×10^5	0.92	0.04	1.00	0.97
	1.00	0.87	1.4×10^5	1.00	1.00	$\mathbf{1.4} \times 10^2$	1.00	1.00	7.7×10^2	0.64	0.00	2.0×10^5	0.92	0.04	1.00	0.96
F5(3D)	0.61	0.00	4.0×10^5	0.98	0.03	4.0×10^5	1.00	1.00	$\mathbf{2.3} \times 10^5$	0.31	0.00	4.0×10^5	-	-	0.88	0.00
	0.46	0.00	4.0×10^5	0.98	0.01	4.0×10^5	1.00	1.00	$\mathbf{2.4} \times 10^5$	0.31	0.00	4.0×10^5	-	-	0.87	0.00
	0.21	0.00	4.0×10^5	0.97	0.01	4.0×10^5	1.00	1.00	$\mathbf{3.0} \times 10^5$	0.28	0.00	4.0×10^5	0.58	0.00	0.88	0.00
	0.07	0.00	4.0×10^5	0.97	0.00	4.0×10^5	**1.00**	**0.86**	3.7×10^5	0.23	0.00	4.0×10^5	0.57	0.00	0.88	0.00
$F6_{(3,4)}$	1.00	1.00	3.1×10^3	1.00	1.00	$\mathbf{1.5} \times 10^3$	1.00	1.00	2.5×10^2	1.00	0.99	5.4×10^3	-	-	1.00	1.00
	1.00	1.00	6.0×10^3	1.00	1.00	$\mathbf{1.6} \times 10^3$	1.00	1.00	3.9×10^2	1.00	0.98	1.0×10^2	-	-	1.00	1.00
	1.00	1.00	1.1×10^2	1.00	1.00	$\mathbf{1.8} \times 10^3$	1.00	1.00	4.3×10^2	1.00	0.96	2.2×10^2	1.00	1.00	1.00	1.00
	1.00	1.00	1.4×10^2	1.00	1.00	$\mathbf{1.9} \times 10^3$	1.00	1.00	4.4×10^2	0.99	0.90	4.4×10^2	1.00	1.00	1.00	1.00
$F6_{(3,3,3)}$	1.00	1.00	4.5×10^2	1.00	1.00	$\mathbf{1.1} \times 10^2$	1.00	1.00	9.1×10^2	0.96	0.27	3.3×10^5	-	-	1.00	1.00
	1.00	0.99	1.5×10^5	1.00	1.00	$\mathbf{1.1} \times 10^2$	1.00	1.00	1.1×10^5	0.92	0.06	3.9×10^5	-	-	1.00	1.00
	0.99	0.83	2.9×10^5	1.00	1.00	$\mathbf{1.1} \times 10^2$	1.00	1.00	1.1×10^5	0.85	0.00	4.0×10^5	-	-	1.00	1.00
	0.96	0.37	3.8×10^5	1.00	1.00	$\mathbf{1.2} \times 10^2$	1.00	1.00	1.1×10^5	0.78	0.00	4.0×10^5	-	-	1.00	1.00
$F6_{(2,\cdots,2)}$	0.48	0.00	4.0×10^5	1.00	1.00	$\mathbf{4.5} \times 10^2$	1.00	1.00	1.3×10^5	0.92	0.05	3.9×10^5	-	-	1.00	1.00
	0.03	0.00	4.0×10^5	1.00	1.00	$\mathbf{4.5} \times 10^2$	1.00	1.00	1.6×10^5	0.83	0.00	4.0×10^5	-	-	1.00	1.00
	0.02	0.00	4.0×10^5	1.00	1.00	$\mathbf{4.7} \times 10^2$	1.00	1.00	1.6×10^5	0.71	0.00	4.0×10^5	-	-	1.00	1.00
	0.02	0.00	4.0×10^5	1.00	1.00	$\mathbf{4.8} \times 10^2$	1.00	1.00	1.7×10^5	0.65	0.00	4.0×10^5	-	-	1.00	1.00
F7(1-2D)	0.57	0.00	2.0×10^5	1.00	1.00	$\mathbf{2.8} \times 10^3$	1.00	0.98	1.0×10^5	0.99	0.94	9.1×10^2	-	-	1.00	1.00
	0.42	0.00	2.0×10^5	1.00	1.00	$\mathbf{2.9} \times 10^3$	0.99	0.91	1.2×10^5	0.98	0.85	1.3×10^5	-	-	1.00	1.00
	0.33	0.00	2.0×10^5	1.00	1.00	$\mathbf{3.0} \times 10^3$	0.94	0.69	1.7×10^5	0.95	0.67	1.6×10^5	1.00	1.00	1.00	1.00
	0.32	0.00	2.0×10^5	1.00	1.00	$\mathbf{3.0} \times 10^3$	0.72	0.03	2.0×10^5	0.92	0.53	1.7×10^5	1.00	1.00	1.00	1.00
F7(2-2D)	0.60	0.00	2.0×10^5	1.00	1.00	$\mathbf{1.4} \times 10^2$	0.98	0.86	1.3×10^5	0.97	0.77	9.1×10^2	-	-	1.00	1.00
	0.31	0.00	2.0×10^5	1.00	1.00	$\mathbf{1.6} \times 10^2$	0.98	0.84	1.5×10^5	0.95	0.56	1.3×10^5	-	-	1.00	1.00
	0.25	0.00	2.0×10^5	1.00	1.00	$\mathbf{1.7} \times 10^2$	0.96	0.66	1.7×10^5	0.96	0.70	1.2×10^5	1.00	1.00	1.00	1.00
	0.20	0.00	2.0×10^5	1.00	1.00	$\mathbf{1.9} \times 10^2$	0.93	0.52	1.9×10^5	0.96	0.67	1.3×10^5	1.00	1.00	1.00	1.00
F7(3-2D)	0.44	0.00	2.0×10^5	0.96	0.77	1.4×10^2	0.96	0.73	1.4×10^5	0.71	0.00	2.0×10^5	-	-	**0.99**	**0.95**
	0.25	0.00	2.0×10^5	0.94	0.65	1.6×10^2	0.91	0.47	1.8×10^5	0.69	0.00	2.0×10^5	-	-	**1.00**	**0.97**
	0.21	0.00	2.0×10^5	0.92	0.56	1.7×10^2	0.65	0.00	2.0×10^5	0.67	0.00	2.0×10^5	0.99	0.92	**0.99**	**0.95**
	0.20	0.00	2.0×10^5	0.92	0.57	1.9×10^2	0.65	0.00	2.0×10^5	0.67	0.00	2.0×10^5	0.99	0.92	**0.99**	**0.94**
F7(3-3D)	0.27	0.00	4.0×10^5	0.68	0.01	4.0×10^5	**0.80**	0.00	4.0×10^5	0.67	0.00	4.0×10^5	-	-	0.75	0.03
	0.26	0.00	4.0×10^5	0.68	0.00	4.0×10^5	**0.73**	0.00	4.0×10^5	0.67	0.00	4.0×10^5	-	-	**0.73**	0.03
	0.30	0.00	4.0×10^5	0.67	0.00	4.0×10^5	0.67	0.00	4.0×10^5	0.67	0.00	4.0×10^5	0.78	0.04	**0.74**	0.02
	0.28	0.00	4.0×10^5	0.68	0.00	4.0×10^5	0.67	0.00	4.0×10^5	0.67	0.00	4.0×10^5	0.77	0.02	**0.74**	0.03

[1] The data of DIDE are from [36]. "-" means these data are not presented in the paper.

As discussed in Section 3, if the PAR-MMO method is used alone, as the ε value decreases, the average number of evaluations required to find all the optima increases a lot, or the percentage of global optima found reduces rapidly. However, if the PAR-MMO method is applied to detect the promising areas and local search is applied to improve the accuracy level of the obtained optima, i.e., the PARL-MMO method, the evaluation budget

can be saved significantly, especially when the ε value is small. When the PARL-MMO method is applied, the number of evaluations used in the PAR-MMO method is not much different, regardless of the ε value, because the definition of the nonpartitionable region is the same. The required total budget size differs depending on the budget required in the local search stage, which is less affected by the ε value compared to the PAR-MMO method.

In most cases, the PARL-MMO method behaves better than the three state-of-the-art methods, except for problems F4(3D), F5(3D), F7(3-2D), and F7(3-3D). In Problem F5(3D), the MOMMOP method has the best performance. Although the PARL-MMO method does not have good performance in this problem according to the criterion SR, the PR is still high (not less than 0.97 for all ε values). For the LAMS-ACO method, a large ant size is required to generate sufficient subpopulations, especially when the number of optima is high. For example, poor performance is observed for Problem F5(3D), because an ant size of 300 is too small for 216 optima. However, a too-large ant size may result in waste of budget. Prior knowledge about the number of optima is important for methods that divide the whole population into subpopulations, whereas this knowledge is not needed in the proposed method.

In Problem F4(3D), the criterion SR of the PARL-MMO method is not as good as that of the EMO-MMO method, but almost all the global optima are discovered (the PR values are all close to one). In addition, a lot of local optima are also contained in the $\mathbb{S}^{\text{opt}}$ in the PARL-MMO method. In problems F7(3-2D) and F7(3-3D), there are numerous local optima that are very close to one of the global optima. For example, in Problem F7(3-2D), the distance between a global optimum and one of the local optima with objective function value 0.5761 is 3×10^{-6}. In this case, if the size of the nonpartitionable region is not small enough to separate these optimal solutions, Algorithm 5 may be stuck in one of the local optima, i.e., the results of PARL-MMO. However, if the size of the nonpartitionable region is small enough, the maximum budget size is not sufficient to reach the corresponding nonpartitionable region, i.e., the results of PAR-MMO. In the EMO-MMO method, the CSO method, which has the ability to escape the local optima, is applied to locally search the promising areas. Therefore, good performance is observed for the EMO-MMO method in problems F7(3-2D) and F7(3-3D).

As for the composition functions with more than five dimensions in the CEC'2013 functions, the proposed method does not have good performance, although all multimodal optimization methods hardly find all global optima in these functions within the given budget. The proposed method has difficulty handling functions with large jumps everywhere. In this case, an intelligent partitioning strategy developed from the system knowledge would be required to make the function smoother.

In summary, the PARL-MMO method has good performance in most of the benchmark functions for capturing multiple global optimal solutions.

4.3. Effect of Local Optima

The PARL-MMO method is applied to functions F8 (one-dimensional increasing optima) and F9 (Rastrigin function), which have only one global optimum and multiple local optima with different qualities, i.e., different objective function values. In this section, the accuracy level ε is set as 1×10^{-4} and 500 replications are executed. Figure 8 shows the frequencies of different optima being found as the total budget size grows. In the Rastrigin function, similar performance is observed for optimal solutions having the same objective function values due to the symmetry property. Thus, to make the figure clear, they are combined as a single line. In the legend, the number in bold shows the number of optimal solutions having this objective function value. Many other local optimal solutions of the Rastrigin function with worse objective function values are not all found due to the budget limitation. Thus, they are not plotted in Figure 8.

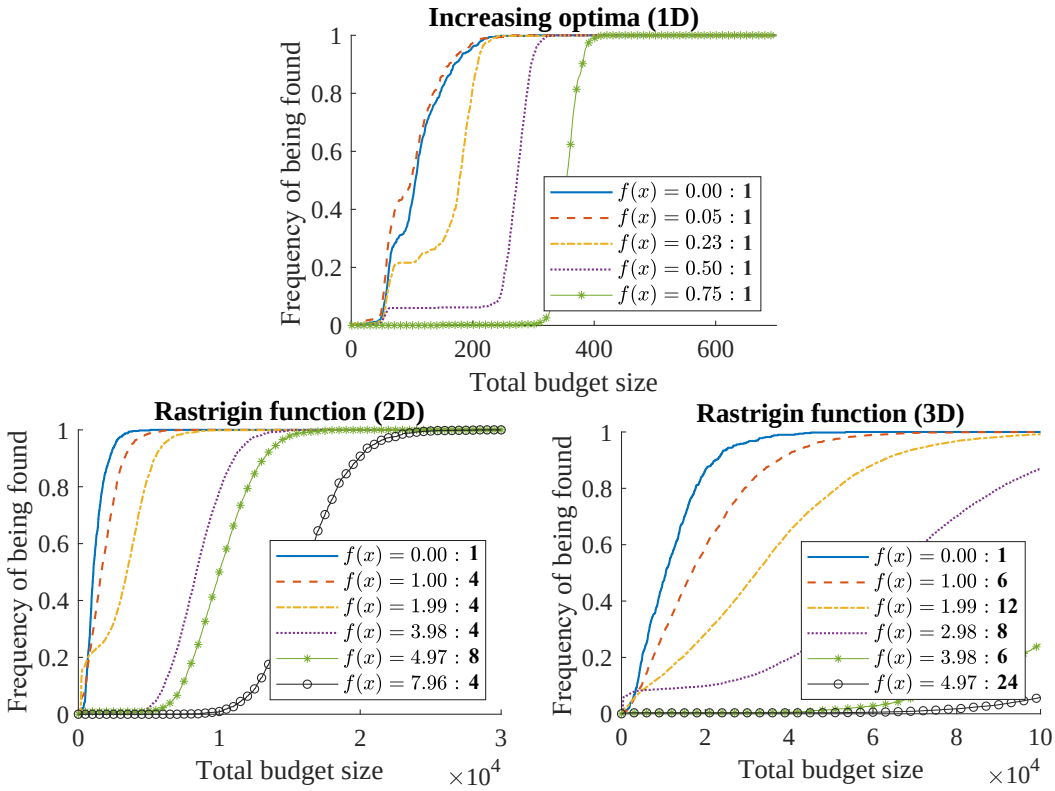


Figure 8. The frequencies of local optima of different qualities being found as the total budget size increases. The number in bold shows the number of optima having this objective function value. A total of 500 replications are executed.

It can be found from Figure 8 that, in the studied cases, the global optimum and the high-quality local optima have higher frequencies to be found than low-quality local optima when the available budget size is small. As the total budget size increases, the rest of the local optima will be found subsequently according to their objective function values. This is one of the features of the proposed method. Other multimodal optimization methods either cannot store local optima, e.g., MOMMOP [59], LAM-ACO [46], and EMO-MMO [68], or optima with different qualities are considered equally important, e.g., [55].

4.4. Schaffer's Function

In this section, the PARL-MMO method is applied to Problem F10, in which the local optimal solution is not a single point. There is only one global optima $f(\mathbf{0}) = 0$ and there are several regions in which all solutions are local optimal solutions with slightly worse objective function values. The optimal solutions located in the same circle have the same objective function values, which are 0.0097, 0.0372, 0.0782, and 0.1270 from inner circle to outer circle, respectively.

Figure 9 shows the partition states and the obtained optima, i.e., the solutions in $\mathbb{S}^{\text{opt}}$, as the total budget size N increases. Unlike EA-based methods, the number of optimal solutions contained in the optimal solution set provided by the proposed method can grow without limit. Multiple local optimal solutions can be captured and the density of the solutions is affected by the size of the nonpartitionable regions and the distance parameter r in Algorithm 3. As discussed in the previous section, the inner circles are found earlier than outer circles.

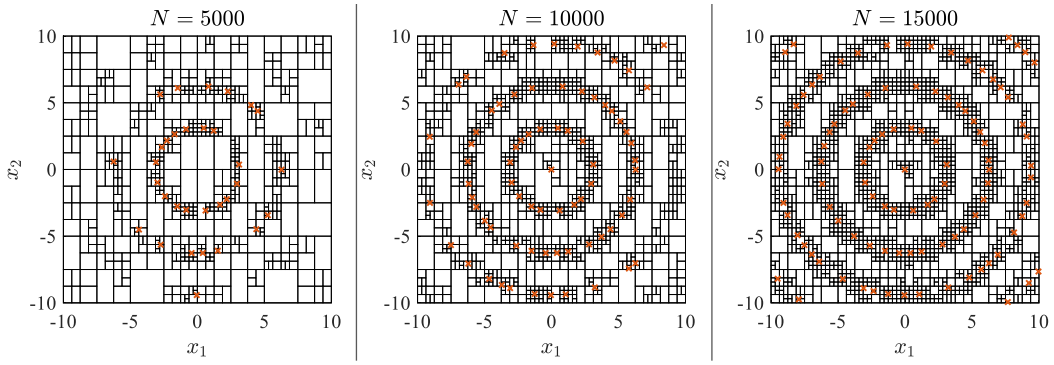


Figure 9. The partition states (a square indicates a region) and the obtained optima (the red crosses) as the total budget size N increases for the Schaffer's function minimization problem.

It should be noticed that if the optimal area is flat in all the dimensions, the variances of all regions (broad and partitionable) within this area will be zero. These regions will not reach the smallest size, since zero weights are assigned according to Equation (5). Thus, the solutions within these regions are not considered as potential optimal solutions in the algorithm. In this situation, an archive can be created to store all the partitionable regions with a good mean and a very low variance.

4.5. Computational Time

In this section, the PAR-MMO method, in which local search is not included, is applied to Problem $F4_{(2,\cdots,2)}$ (five dimensions) with $\varepsilon = 1 \times 10^{-4}$. A total of 20 independent replications are executed in Matlab R2020a on a computer (Intel(R) Core(TM) i7-7700U CPU @ 3.6 GHz and 16 GB of RAM). Figure 10 shows how the average computational time changes as the total budget size increases. It can be found that the computational time increases quadratically as discussed in Section 2.3 ($R^2 = 100.0\%$ for the quadratic regression). Although the proposed method will become time-consuming when the total budget size is very large, it is still efficient in the studied case (on average, 121 s for a budget of size 1×10^6) and the computational time climbs slowly before the total budget size reaches one million.

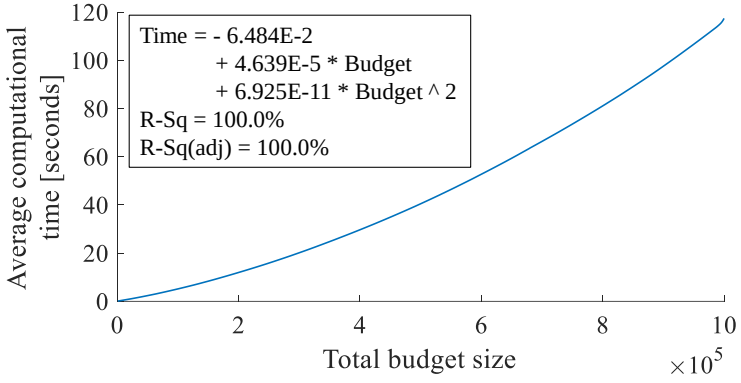


Figure 10. The average computational time required for the proposed method in Problem $F4_{(2,\cdots,2)}$ as the total budget increases. A total of 20 replications are executed.

5. Application

The optimal control problem of a nonlinear stirred tank reactor [72] is considered in this section. The chemical process can be modeled by the following differential equations:

$$\dot{x}_1 = -(2+u)(x_1+0.25) + (x_2+0.5)\exp\left(\frac{25x_1}{x_1+2}\right), \tag{8}$$

$$\dot{x}_2 = 0.5 - x_2 - (x_2 + 0.5)\exp\left(\frac{25x_1}{x_1 + 2}\right), \tag{9}$$

$$\dot{x_3} = x_1^2 + x_2^2 + 0.1u^2. \tag{10}$$

where $u(t), u(t) \in [0.0, 5.0]$ is the flow rate of the cooling fluid, $x_1(t)$ is the dimensionless steady temperature, $x_2(t)$ is the deviation from dimensionless steady concentration, and the interval of integration is $0 \leq t \leq 0.78$. The performance index to minimize is $f = x_3(0.78)$, which can be evaluated using *ode45* in Matlab, and the initial condition is $x_1(0) = 0.09, x_2(0) = 0.09, x_3(0) = 0$. The problem has a global optimum $x_3(0.78) = 0.13309$ and a local optimum $x_3(0.78) = 0.24442$. These two values are directly associated with two different control trajectories [72].

To solve the problem, the time interval is discretized into 13 time slots in order to obtain a reasonably good integration accuracy, and constant control is used within a time slot. Then, $u(t)$ becomes 13 decision variables $[u(1), u(2), \cdots, u(13)]^T$. The PARLMMO method is applied with $\alpha = 0.3, n_0 = 3, n_{\max} = 8, \Delta = 3$. The edge threshold that is considered as nonpartitionable is set to 0.8 and the stop threshold of the local search is set as 0.01. The maximum budget size is 1×10^5.

Twenty replications are carried out. The optimum is regarded as found only when the algorithm recognizes that the point is an optimum and saves it in $\mathbb{S}^{\text{opt}}$. Table 4 shows the number of the replications that find each optimum, the average budget to find each optimum, and the average absolute percentage gap of the objective function value compared to the real optimum (0.13309 and 0.24442). It should be noticed that the gap could be caused by the precision of the captured solution, the calculation of the integration, and the discretization of $u(t)$.

Table 4. The results of the nonlinear stirred tank reactor control problem (20 replications).

PARL-MMO	Successful Replications	Average Budget	Average Absolute Percentage Gap
Global optimum	20	34,156	1.8%
Local optimum	3	771	0.5%
MOMMOP	**Successful Replications**	**Average Budget**	**Average Absolute Percentage Gap**
Global optimum	20	-	1.4%
Local optimum	0	-	-

For comparison purpose, the MOMMOP method [59], which outperforms other methods in the benchmark functions, is also applied with population size 30, mutation factor 0.5, crossover index 0.7, crowded radius 0.01, and maximum budget size 1×10^5. The optimal solution is considered as captured if it is contained in the Pareto set.

The first comment is that the global optimum is captured in all replications. In addition, no points other than these two optimal solutions appear in the final optimization set. This means that the algorithm will not mislead the users with fake optima.

However, the local optimum is captured only in three replications. This is because the local optimum is 84% worse than the global optimum. Thus, the area around the local optimum is not considered as promising by the algorithm, unless other areas with better objective function values have not been discovered or have been exploited. In this case, according to the budget size required to find each optimum, the area around the local optimum is only searched before the area around the global optimum is discovered. This result is consistent with the feature of the algorithm that focuses on the global optima and the high-quality local optima. When only good solutions are of interest, the algorithm would not waste budget to search for optima with poor objective functions. However, this may become a disadvantage when all optima (no matter if good or poor) are required.

The MOMMOP method is proposed for seeking multiple global optimal solutions. Thus, all points in the Pareto set are points around the global optimum, and the local optimum is not identified in any replication. The best solution in the Pareto set is used to

calculate the average absolute percentage gap. This gap is less than the PARL-MMO. A possible reason for this is that the function around the global optimum is not smooth due to the discretization of the integration. Therefore, the local search may be trapped when refining the found solution in the PARL-MMO method. This may be improved by using different algorithms, such as EA, in the refining phase.

6. Conclusions

A partition-based random search method is proposed, in which by controlling the partition rates of different regions, promising areas are exploited probabilistically earlier than nonpromising areas. Multiple optimal solutions (both global and local) can be found and stored. It does not require prior knowledge about the number of optima in the studied problem and it is not sensitive to the distance parameter.

Numerical results show that, by cooperating with local search, the proposed method has good performance in finding multiple global optimal solutions in 14 benchmark functions compared to four state-of-the-art methods. In problems containing local optima of different qualities, i.e., different objective function values, high-quality local optima will be found earlier than low-quality optima. Therefore, it will focus on exploiting global optima and high-quality local optima when the budget is not quite sufficient. In addition, the proposed method can also deal with optimization problems that exist in regions where all solutions are optimal solutions.

One of the limitations of the proposed method is that a large amount of calculation memory is needed because all existing regions and all sampled solutions are stored. The increased number of regions also makes the computational time increase quadratically as the total budget size increases, although it is still efficient when the total budget size is not extremely large. Therefore, the pruning of the stored regions is one of the directions of further work. The other limitation is introduced by the property of the partition-based random search framework. Partition-based methods are efficient for problems in which each decision variable has a large search space. However, it could be inefficient for high-dimensional problems if the partition strategy is simply partitioning each dimension into several ranges. In this case, intelligent partition strategies should be developed based on the features of the studied problem to improve the efficiency of the algorithm. The efficiency of the proposed method could be highly affected by selection of the partition strategy.

The future development includes several directions. The first one is the development of an algorithm for the deletion of less interesting regions to solve the problem of quadratically increased computational time. The second one is adopting an adaptive α value in the proposed method. As discussed in the paper, a low α value focuses more on exploration, whereas a high α value focuses more on exploitation. Therefore, an increasing α value may improve the efficiency of the proposed method. The third one is to deal with the optimization problems with constraints. Although the constraints can be handled by manipulating the sampling strategy to avoid the sampling of infeasible solutions, this action may be difficult for some applications. Penalty function is another common way to deal with constraints, but the regions containing the boundary may have biased performance estimates, which may affect the proposed method. Therefore, an adaptive penalty function to deal with the constraints in the proposed method will be an interesting research direction. The last one is extending the method to optimization problems with stochastic objective function or constraints.

Author Contributions: All the authors contributed to the conceptualization of this research. Z.L. also contributed to the development of the methodology, software coding, draft writing, and validation of results. A.M. also contributed to the supervision of the research, development of the methodology and validation of results. S.D. also ncontributed to the validation of results, funding acquisition and administration. E.S. also contributed to the validation of results and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (Grant No. 52275499) and the National key R&D plan of China (No. 2022YFF0605700).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Selected Benchmark Functions

The functions considered for numerical experiments are here summarized and show in Figure A1.

F1: One-dimensional equal optima

$f(x) = 1 - \sin^6(5\pi x), x \in [0, 1]$.

Property: Five global optima evenly distributed: $x_i^* = 0.2i - 0.1$ that $f(x_i^*) = 0$.

F2: Himmelblau's function

$f(\boldsymbol{x}) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2, x_i \in [-6, 6], i = 1, 2.$

Property: Four global optima with two closer to each other: $\boldsymbol{x}_1^* = (3.0, 2.0)$, $\boldsymbol{x}_2^* = (-2.805118, 3.131312)$, $\boldsymbol{x}_3^* = (-3.779310, -3.283186)$ and $\boldsymbol{x}_4^* = (3.584428, -1.848126)$ that $f(\boldsymbol{x}_i^*) = 0$.

F3: Six-hump camel back

$f(\boldsymbol{x}) = 4((4 - 2.1x_1^2 + x_1^4/3)x_1^2 + x_1x_2 + (4x_2^2 - 4)x_2^2), x_1 \in [-1.9, 1.9], x_2 \in [-1.1, 1.1].$

Property: Two global optima: $\boldsymbol{x}_1^* = (0.0898, -0.7126)$ and $\boldsymbol{x}_2^* = (-0.0898, 0.7126)$ that $f(\boldsymbol{x}_i^*) = -4.1265$. Four local optima: $\boldsymbol{x}_3^* = (1.7035, -0.7961)$ and $\boldsymbol{x}_4^* = (-1.7035, 0.7961)$ that $f(\boldsymbol{x}_i^*) = -0.8619$ and $\boldsymbol{x}_5^* = (-1.6071, -0.5687)$ and $\boldsymbol{x}_6^* = (1.6071, 0.5687)$ that $f(\boldsymbol{x}_i^*) = 8.4170$.

F4: Shubert function

$f(\boldsymbol{x}) = \prod_{i=1}^{d} \sum_{j=1}^{5} j\cos((j+1)x_i + j), x_i \in [-10, 10], \forall i.$

Property: $d \cdot 3^d$ global optima in 3^d groups with d optima in each group and many poor local optima. The global optima are $f(\boldsymbol{x}_i^*) = -186.7309$ for two dimensions and $f(\boldsymbol{x}_i^*) = -2709.0935$ for three dimensions.

F5: Vincent function

$f(\boldsymbol{x}) = 1 - \frac{1}{d}\sum_{i=1}^{d} \sin(10\log(x_i)), x_i \in [0.25, 10], \forall i.$

Property: 6^d global optima unevenly distributed: $\boldsymbol{x}_{i_1,\cdots,i_d}^* = \left(\exp\left(\frac{(4i_1-11)\pi}{20}\right), \cdots, \exp\left(\frac{(4i_d-11)\pi}{20}\right)\right)$ that $f(\boldsymbol{x}_i^*) = 0$.

F6: Modified Rastrigin function $(k_1, \cdots, k_d)$

$f(\boldsymbol{x}) = \sum_{i=1}^{d}(10 + 9\cos(2\pi k_i x_i)), x_i \in [0, 1], \forall i.$

Property: $\prod_{i=1}^{d} k_i$ global optima evenly distributed: $\boldsymbol{x}_{i_1,\cdots,i_d}^* = \left(\frac{2i_1-1}{2k_1}, \cdots, \frac{2i_d-1}{2k_d}\right)$ that $f\left(\boldsymbol{x}_{i_1,\cdots,i_d}^*\right) = d$.

F7: Composition function

Property: A multimodal function composed of several basic functions; thus, different functions' properties are mixed together. More details can be found in [69]. The label "F7(a-dD)" in the paper indicates the composition function a with d dimension.

F8: One-dimensional increasing optima

$f(x) = 1 - \exp\left(-2\log(2)\left(\frac{x-0.08}{0.854}\right)^2\right)\sin^6(5\pi(x^{3/4} - 0.05)), x \in [0.02, 1].$

Property: One global optimum and four increasing local optima.

F9: Rastrigin function

$f(\boldsymbol{x}) = \sum_{i=1}^{d}(x_i^2 - 10\cos(2\pi x_i) + 10), x_i \in [-5.12, 5.12], \forall i.$

Property: One global optimum: $f(\mathbf{0}) = 0$, and $11^d - 1$ local optima with different qualities.

F10: Schaffer's function

$$f(\boldsymbol{x}) = 0.5 + \frac{\sin^2\left(\sqrt{\sum_{i=1}^{d} x_i^2}\right) - 0.5}{(1+0.001(\sum_{i=1}^{d} x_i^2))^2}, x_i \in [-10, 10], \forall i.$$

Property: One global optimum $f(\mathbf{0}) = 0$ and seven regions where all solutions are local optimal.

Appendix B. Algorithm Parameter

The regions are considered as nonpartitionable regions when all edges are smaller than values in Table A1 of Appendix B.

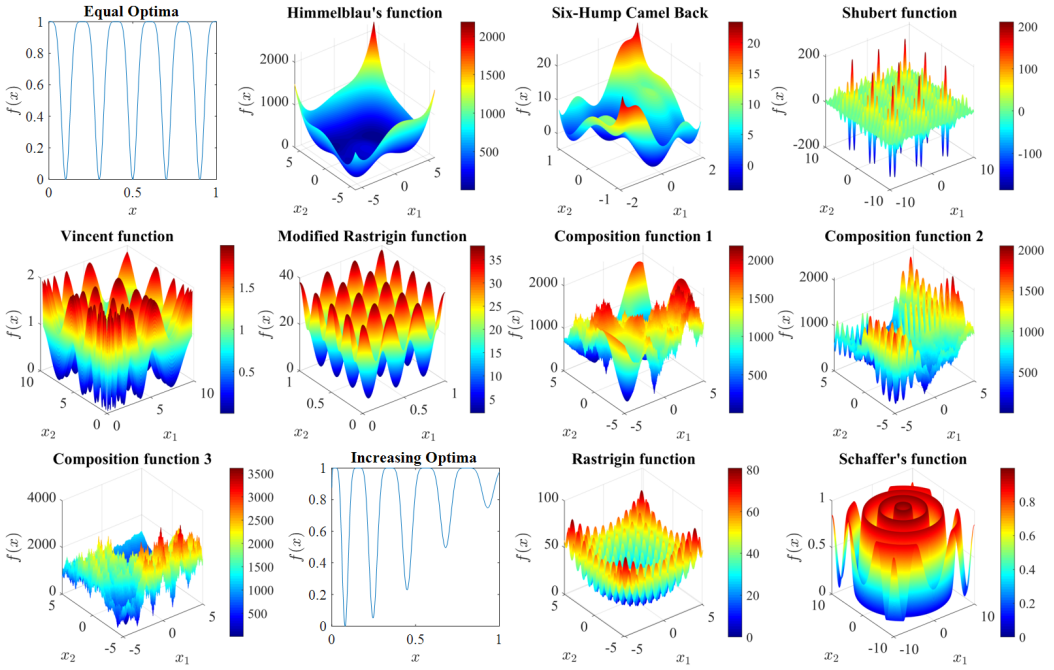


Figure A1. The surface plots of the selected benchmark functions. f_4, f_5, f_6, f_7, f_9 only show 2D cases.

Table A1. The edge threshold of nonpartitionable regions.

Func \ ε	0.1	0.01	0.001	0.0001	Local Search
F1	2.3×10^{-2}	7.5×10^{-3}	2.3×10^{-3}	7.5×10^{-4}	4.0×10^{-2}
F2	7.4×10^{-2}	2.4×10^{-2}	7.4×10^{-3}	2.4×10^{-3}	4.0×10^{-1}
F3	8.5×10^{-2}	2.8×10^{-2}	8.5×10^{-3}	2.8×10^{-3}	1.5×10^{-1}
F4(2D)	9.6×10^{-3}	3.0×10^{-3}	9.6×10^{-4}	3.0×10^{-4}	1.6×10^{-1}
F4(3D)	2.0×10^{-3}	6.4×10^{-4}	2.0×10^{-4}	6.4×10^{-5}	1.6×10^{-1}
F5(2D)	3.0×10^{-2}	9.6×10^{-3}	3.0×10^{-3}	9.6×10^{-4}	8.0×10^{-2}
F5(3D)	3.0×10^{-2}	9.6×10^{-3}	3.0×10^{-3}	9.6×10^{-4}	8.0×10^{-2}
$F6_{(3,4)}$	9.6×10^{-3}	3.0×10^{-3}	9.6×10^{-4}	3.0×10^{-4}	4.0×10^{-2}
$F6_{(3,3,3)}$	9.0×10^{-3}	3.0×10^{-3}	9.0×10^{-4}	3.0×10^{-4}	4.0×10^{-2}
$F6_{(2,\cdots,2)}$	1.1×10^{-2}	3.4×10^{-3}	1.1×10^{-3}	3.4×10^{-4}	7.0×10^{-2}
F7(1-2D)	5.0×10^{-7}	1.6×10^{-8}	1.1×10^{-9}	3.2×10^{-1}	1.6×10^{-1}
F7(2-2D)	1.6×10^{-6}	5.6×10^{-8}	1.8×10^{-9}	5.5×10^{-1}	1.6×10^{-1}
F7(3-2D)	9.5×10^{-8}	2.0×10^{-9}	1.5×10^{-1}	4.3×10^{-1}	2.0×10^{-2}
F7(3-3D)	9.0×10^{-8}	2.5×10^{-9}	1.4×10^{-1}	3.8×10^{-1}	2.0×10^{-2}
F8	1.7×10^{-2}	5.2×10^{-3}	1.7×10^{-3}	5.2×10^{-4}	2.0×10^{-2}
F9(2D)	3.0×10^{-2}	1.0×10^{-2}	3.0×10^{-3}	1.0×10^{-3}	1.0×10^{-1}
F9(3D)	2.6×10^{-2}	8.0×10^{-3}	2.6×10^{-3}	8.0×10^{-4}	1.0×10^{-1}
F10	4.0×10^{-1}	4.0×10^{-1}	2.0×10^{-1}	8.0×10^{-2}	4.0×10^{-1}

Appendix C. ANOVA

In this Appendix C the main effect plots and the ANOVA tables on functions, where all the global optima could be found within the maximum budget, are presented (as shown in Figure A2). The experimental settings are the same as in Section 4.1. A total of 500 replications are executed and divided into ten batches, except for Problem $F6_{(2,\cdots,2)}$, where 20 replications are executed. Similar to Problem $F6_{(3,3,3)}$, almost all factors have significant effects on the algorithm, except some interactions. The effects of parameter $n_{\max}$ and parameter Δ are larger than the effect of parameter α based on the F-values (except for Problem F7(2-2D)), although the influence of parameter α is comparable to that of parameter Δ on Problem F2. For Problem F7(2-2D), the influences of the three parameters are comparable.

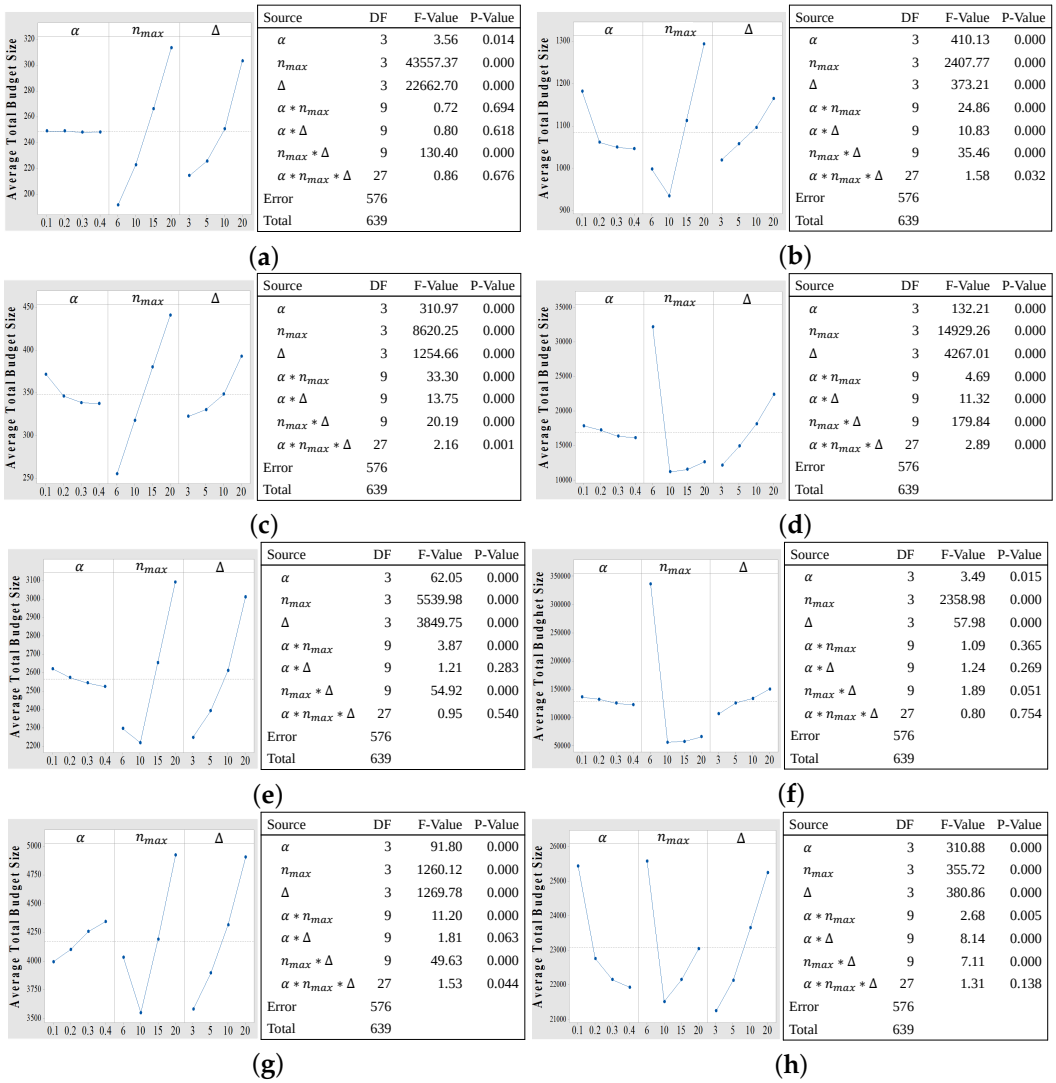


Figure A2. The main effect plots and the ANOVA tables on different multimodal optimization benchmark functions. (**a**) F1: One-dimensional equal optima. (**b**) F2: Himmelblau's function. (**c**) F3: Six-hump camel back. (**d**) F4(2D): Shubert function. (**e**) $F6_{(3,4)}$: Modified Rastrigin function. (**f**) $F6_{(2,\cdots,2)}$: Modified Rastrigin function. (**g**) F7(1-2D): Composition function 1. (**h**) F7(2-2D): Composition function 2.

References

1. Hu, C.; Zhao, J.; Yan, X.; Zeng, D.; Guo, S. A MapReduce based Parallel Niche Genetic Algorithm for contaminant source identification in water distribution network. *Ad Hoc Netw.* **2015**, *35*, 116–126. [CrossRef]
2. Forrester, A.I.; Keane, A.J. Recent advances in surrogate-based optimization. *Prog. Aerosp. Sci.* **2009**, *45*, 50–79. [CrossRef]

3. Shahriari, B.; Swersky, K.; Wang, Z.; Adams, R.P.; De Freitas, N. Taking the human out of the loop: A review of bayesian optimization. *Proc. IEEE* **2016**, *104*, 148–175. [CrossRef]
4. Qu, B.Y.; Liang, J.J.; Wang, Z.; Chen, Q.; Suganthan, P.N. Novel benchmark functions for continuous multimodal optimization with comparative results. *Swarm Evol. Comput.* **2016**, *26*, 23–34. [CrossRef]
5. Li, X.; Epitropakis, M.G.; Deb, K.; Engelbrecht, A. Seeking multiple solutions: An updated survey on niching methods and their applications. *IEEE Trans. Evol. Comput.* **2017**, *21*, 518–538. [CrossRef]
6. Koper, K.D.; Wysession, M.E.; Wiens, D.A. Multimodal function optimization with a niching genetic algorithm: A seismological example. *Bull. Seismol. Soc. Am.* **1999**, *89*, 978–988. [CrossRef]
7. Kronfeld, M.; Dräger, A.; Aschoff, M.; Zell, A. On the benefits of multimodal optimization for metabolic network modeling. In Proceedings of the German Conference on Bioinformatics 2009. Gesellschaft für Informatik eV, Halle (Saale), Germany, 28–30 September 2009.
8. Pérez, E.; Posada, M.; Herrera, F. Analysis of new niching genetic algorithms for finding multiple solutions in the job shop scheduling. *J. Intell. Manuf.* **2012**, *23*, 341–356. [CrossRef]
9. Preuss, M.; Burelli, P.; Yannakakis, G.N. Diversified virtual camera composition. In Proceedings of the European Conference on the Applications of Evolutionary Computation, Malaga, Spain, 11–13 April 2012; Springer: Berlin/Heidelberg, Germany, 2012, pp. 265–274.
10. Kamyab, S.; Eftekhari, M. Feature selection using multimodal optimization techniques. *Neurocomputing* **2016**, *171*, 586–597. [CrossRef]
11. Beasley, D.; Bull, D.R.; Martin, R.R. A sequential niche technique for multimodal function optimization. *Evol. Comput.* **1993**, *1*, 101–125. [CrossRef]
12. Parsopoulos, K.E.; Vrahatis, M.N. On the computation of all global minimizers through particle swarm optimization. *IEEE Trans. Evol. Comput.* **2004**, *8*, 211–224. [CrossRef]
13. Vitela, J.E.; Castaños, O. A sequential niching memetic algorithm for continuous multimodal function optimization. *Appl. Math. Comput.* **2012**, *218*, 8242–8259. [CrossRef]
14. Zhang, J.; Huang, D.S.; Lok, T.M.; Lyu, M.R. A novel adaptive sequential niche technique for multimodal function optimization. *Neurocomputing* **2006**, *69*, 2396–2401. [CrossRef]
15. Li, L.; Hong-Qi, L.; Shao-Long, X. Particle swarm multi_optimizer for locating all local solutions. In Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–6 June 2008, pp. 1040–1046.
16. Srinivas, M.; Patnaik, L.M. Genetic algorithms: A survey. *Computer* **1994**, *27*, 17–26. [CrossRef]
17. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
18. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
19. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
20. Pétrowski, A. A clearing procedure as a niching method for genetic algorithms. In Proceedings of the 3rd IEEE International Conference on Evolutionary Computation (ICEC'96), Nayoya University, Nayoya, Japan, 20–22 May 1996; pp. 798–803.
21. Singh, G.; Deb, K. Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In Proceedings of the 8th Annual Conference on Genetic and eVolutionary Computation, Seattle, WA, USA, 8–12 July 2006; pp. 1305–1312.
22. Goldberg, D.E.; Richardson, J. Genetic algorithms with sharing for multimodal function optimization. In Proceedings of the Second International Conference on Genetic algorithms and Their Applications, Hillsdale, NJ, USA, 28–31 July 1987; pp. 41–49.
23. Deb, K.; Goldberg, D.E. An investigation of niche and species formation in genetic function optimization. In Proceedings of the Third International Conference on Genetic Algorithms, San Francisco, CA, USA, 4–7 June 1989; pp. 42–50.
24. Goldberg, D.E.; Wang, L. Adaptive niching via coevolutionary sharing. *Genet. Algorithms Evol. Strategy Eng. Comput. Sci.* **1997**, *97007*, 21–38.
25. MacQueen, J. Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, University of California, Berkeley, CA, USA, 21 June–18 July 1965; Volume 1, pp. 281–297.
26. Yin, X.; Germay, N. A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization. In Proceedings of the Artificial Neural Nets and Genetic Algorithms, Innsbruck, Austria, 14–16 April 1993; pp. 450–457.
27. Mahfoud, S.W. Crowding and preselection revisited. In Proceedings of the PPSN, Amsterdam, The Netherlands, April 1992; Volume 2, pp. 27–36.
28. Mengshoel, O.J.; Goldberg, D.E. Probabilistic crowding: Deterministic crowding with probabilistic replacement. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO1999), Orlando, FL, USA, 13–17 July 1999; Volume 1, pp. 409–416.
29. Harik, G.R. Finding Multimodal Solutions Using Restricted Tournament Selection. In Proceedings of the ICGA, Pittsburgh, PA, USA, 15–19 July 1995; pp. 24–31.

30. Li, J.P.; Balazs, M.E.; Parks, G.T.; Clarkson, P.J. A species conserving genetic algorithm for multimodal function optimization. *Evol. Comput.* **2002**, *10*, 207–234. [CrossRef]
31. Li, J.P.; Wood, A. Random search with species conservation for multimodal functions. In Proceedings of the 2009 IEEE Congress on Evolutionary Computation, Trondheim, Norway, 18–21 May 2009; pp. 3164–3171.
32. Li, X. Niching without niching parameters: Particle swarm optimization using a ring topology. *IEEE Trans. Evol. Comput.* **2010**, *14*, 150–169. [CrossRef]
33. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the Sixth International, Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995; pp. 39–43.
34. Das, S.; Maity, S.; Qu, B.Y.; Suganthan, P.N. Real-parameter evolutionary multimodal optimization—A survey of the state-of-the-art. *Swarm Evol. Comput.* **2011**, *1*, 71–88. [CrossRef]
35. Zhao, H.; Zhan, Z.H.; Lin, Y.; Chen, X.; Luo, X.N.; Zhang, J.; Kwong, S.; Zhang, J. Local binary pattern-based adaptive differential evolution for multimodal optimization problems. *IEEE Trans. Cybern.* **2019**, *50*, 3343–3357. [CrossRef]
36. Chen, Z.G.; Zhan, Z.H.; Wang, H.; Zhang, J. Distributed individuals for multiple peaks: A novel differential evolution for multimodal optimization problems. *IEEE Trans. Evol. Comput.* **2019**, *24*, 708–719. [CrossRef]
37. Wang, Z.J.; Zhan, Z.H.; Lin, Y.; Yu, W.J.; Wang, H.; Kwong, S.; Zhang, J. Automatic niching differential evolution with contour prediction approach for multimodal optimization problems. *IEEE Trans. Evol. Comput.* **2020**, *24*, 114–128. [CrossRef]
38. Tsutsui, S.; Fujimoto, Y.; Ghosh, A. Forking genetic algorithms: GAs with search space division schemes. *Evol. Comput.* **1997**, *5*, 61–80. [CrossRef] [PubMed]
39. Ursem, R.K. Multinational evolutionary algorithms. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 6–9 July 1999; Volume 3, pp. 1633–1640.
40. Siarry, P.; Pétrowski, A.; Bessaou, M. A multipopulation genetic algorithm aimed at multimodal optimization. *Adv. Eng. Softw.* **2002**, *33*, 207–213. [CrossRef]
41. Brits, R.; Engelbrecht, A.P.; Bergh, F.V.D. A niching particle swarm optimizer. In Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution And Learning, Singapore, 18–22 November 2002; Volume 2.
42. Parrott, D.; Li, X. Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Trans. Evol. Comput.* **2006**, *10*, 440–458. [CrossRef]
43. Wang, J.; Xie, Y.; Xie, S.; Chen, X. Cooperative particle swarm optimizer with depth first search strategy for global optimization of multimodal functions. *Appl. Intell.* **2022**, *52*, 10161–10180. [CrossRef]
44. Lin, X.; Luo, W.; Xu, P.; Qiao, Y.; Yang, S. PopDMMO: A general framework of population-based stochastic search algorithms for dynamic multimodal optimization. *Swarm Evol. Comput.* **2022**, *68*, 101011. . [CrossRef]
45. Alami, J.; El Imrani, A.; Bouroumi, A. A multipopulation cultural algorithm using fuzzy clustering. *Appl. Soft Comput.* **2007**, *7*, 506–519. [CrossRef]
46. Yang, Q.; Chen, W.N.; Yu, Z.; Gu, T.; Li, Y.; Zhang, H.; Zhang, J. Adaptive multimodal continuous ant colony optimization. *IEEE Trans. Evol. Comput.* **2017**, *21*, 191–205. [CrossRef]
47. Wang, Z.J.; Zhan, Z.H.; Lin, Y.; Yu, W.J.; Zhang, J. Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems. *IEEE Trans. Evol. Comput.* **2018**, *22*, 894–908. [CrossRef]
48. Bird, S.; Li, X. Adaptively choosing niching parameters in a PSO. In Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, Seattle, WA, USA, 8–12 July 2006; pp. 3–10.
49. Shir, O.M.; Bäck, T. Niche radius adaptation in the cma-es niching algorithm. In *Parallel Problem Solving from Nature-PPSN IX*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 142–151.
50. Nayak, S.; Kar, S.K.; Dash, S.S.; Vishnuram, P.; Thanikanti, S.B.; Nastasi, B. Enhanced Salp Swarm Algorithm for Multimodal Optimization and Fuzzy Based Grid Frequency Controller Design. *Energies* **2022**, *15*, 3210. . [CrossRef]
51. Yao, J.; Kharma, N.; Zhu, Y.Q. On clustering in evolutionary computation. In Proceedings of the 2006 IEEE International Conference on Evolutionary Computation, Vancouver, BC, Canada, 16–21 July 2006; pp. 1752–1759.
52. Wessing, S.; Preuss, M.; Rudolph, G. Niching by multiobjectivization with neighbor information: Trade-offs and benefits. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, 20–23 June 2013; pp. 103–110.
53. Yao, J.; Kharma, N.; Grogono, P. BMPGA: A bi-objective multi-population genetic algorithm for multi-modal function optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, Edinburgh, UK, 2–4 September 2005; Volume 1, pp. 816–823.
54. Yao, J.; Kharma, N.; Grogono, P. Bi-objective multipopulation genetic algorithm for multimodal function optimization. *IEEE Trans. Evol. Comput.* **2010**, *14*, 80–102.
55. Deb, K.; Saha, A. Finding multiple solutions for multimodal optimization problems using a multi-objective evolutionary approach. In Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, Portland, OR, USA, 7–11 July 2010; pp. 447–454.
56. Deb, K.; Saha, A. Multimodal optimization using a bi-objective evolutionary algorithm. *Evol. Comput.* **2012**, *20*, 27–62. [CrossRef] [PubMed]
57. Basak, A.; Das, S.; Tan, K.C. Multimodal optimization using a biobjective differential evolution algorithm enhanced with mean distance-based selection. *IEEE Trans. Evol. Comput.* **2013**, *17*, 666–685. [CrossRef]
58. Bandaru, S.; Deb, K. A parameterless-niching-assisted bi-objective approach to multimodal optimization. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, 20–23 June 2013; pp. 95–102.

59. Wang, Y.; Li, H.X.; Yen, G.G.; Song, W. MOMMOP: Multiobjective optimization for locating multiple optimal solutions of multimodal optimization problems. *IEEE Trans. Cybern.* **2015**, *45*, 830–843. [CrossRef] [PubMed]
60. Yu, W.J.; Ji, J.Y.; Gong, Y.J.; Yang, Q.; Zhang, J. A tri-objective differential evolution approach for multimodal optimization. *Inf. Sci.* **2018**, *423*, 1–23. [CrossRef]
61. Shi, L.; Olafsson, S. *Nested Partitions Method, Theory and Applications*; Springer: Berlin/Heidelberg, Germany, 2009.
62. Hong, L.J.; Nelson, B.L. Discrete optimization via simulation using COMPASS. *Oper. Res.* **2006**, *54*, 115–129. [CrossRef]
63. Xu, J.; Nelson, B.L.; Hong, L.J. An adaptive hyperbox algorithm for high-dimensional discrete optimization via simulation problems. *INFORMS J. Comput.* **2013**, *25*, 133–146. [CrossRef]
64. Zabinsky, Z.B.; Wang, W.; Prasetio, Y.; Ghate, A.; Yen, J.W. Adaptive probabilistic branch and bound for level set approximation. In Proceedings of the 2011 Winter Simulation Conference (WSC), Phoenix, AZ, USA, 11–14 December 2011; pp. 4146–4157.
65. Zabinsky, Z.B.; Huang, H. A partition-based optimization approach for level set approximation: Probabilistic branch and bound. In *Women in Industrial and Systems Engineering*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 113–155.
66. Lin, Z.; Matta, A.; Du, S. A new partition-based random search method for deterministic optimization problems. In Proceedings of the 2019 Winter Simulation Conference (WSC), National Harbor, MD, USA, 8–11 December 2019; pp. 3504–3515.
67. Lin, Z.; Matta, A.; Du, S. A budget allocation strategy minimizing the sample set quantile for initial experimental design. *IISE Trans.* **2020**, *53*, 39–57. [CrossRef]
68. Cheng, R.; Li, M.; Li, K.; Yao, X. Evolutionary multiobjective optimization-based multimodal optimization: Fitness landscape approximation and peak detection. *IEEE Trans. Evol. Comput.* **2018**, *22*, 692–706. [CrossRef]
69. Li, X.; Engelbrecht, A.; Epitropakis, M.G. *Benchmark Functions for CEC'2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization*; Technical Report; RMIT University, Evolutionary Computation and Machine Learning Group: Melbourne, Australia, 2013.
70. Socha, K.; Dorigo, M. Ant Colony Optimization For Continuous Domains. *Eur. J. Oper. Res.* **2008**, *185*, 1155–1173. [CrossRef]
71. Cheng, R.; Jin, Y. A competitive swarm optimizer for large scale optimization. *IEEE Trans. Cybern.* **2015**, *45*, 191–204. [CrossRef] [PubMed]
72. Ali, M.M.; Storey, C.; Torn, A. Application of stochastic global optimization algorithms to practical problems. *J. Optim. Theory Appl.* **1997**, *95*, 545–563. [CrossRef]