

Adaptive Memetic Differential Evolution with Niching Competition and Supporting Archive Strategies for Multimodal Optimization*

Weiguo Sheng[a,*], Xi Wang[a], Zidong Wang[b], Qi Li[a], Yun Chen[c]

[a] *Department of Computer Science, Hangzhou Normal University, Hangzhou, 311121, China*
[b] *Department of Computer Science, Brunel University London, Uxbridge, Middlesex, UB8 3PH, UK*
[c] *Institute of Information and Control, Hangzhou Dianzi University, Hangzhou 310018, China*

Abstract

Multimodal optimization, which aims at locating multiple optimal solutions within the search space, is inherently a difficult problem. This work proposes an adaptive memetic differential evolution algorithm with niching competition and supporting archive strategies to tackle the problem. In the proposed algorithm, a niching competition strategy is designed to competitively employ niches according to their potentials by encouraging high potential niches for exploitation while low potential niches for exploration, thus appropriately searching the space to identify multiple optima. Further, a supporting archive strategy is devised and implemented at the niche level with a dual purpose of helping maintain potential optima as well as facilitate the evolution of population. In this strategy, the writing and reading of archive is implicitly implemented during evolution rather than requiring external rules. Additionally, an adaptive Cauchy-based local search scheme, which considers the possible locations of optima to implement the local search, is developed and incorporated into the proposed method to efficiently and properly improve niching seeds. The resulting algorithm has

*This work was supported in part by the National Natural Science Foundation of China under Grant 61883082, Grant 62003121 and Zhejiang Provincial Natural Science Foundation of China under Grant LQ20F030014.

*Corresponding author
Email address: `w.sheng@ieee.org` (Weiguo Sheng)

been evaluated with extensive experiments on benchmark functions as well as a robot kinematics problem and compared with related methods. The results show that our method is able to consistently and accurately locate multiple optima in the solution space, and outperform related methods.

Keywords: Differential evolution, multimodal optimization, niching method, archive technique, local search

1. Introduction

Evolutionary algorithms (EAs), a kind of population-based stochastic optimization technique, are typically designed to deliver one single optimal solution of the given optimization problem. However, many real optimization problems could involve multiple optima and it is desirable to simultaneously locate these optima. For instance, for pedestrian detection problem [1], it usually needs to find multiple pedestrian routes from an image, so that the user could make the decision based on his/her preference. Other examples include job scheduling [2], electromagnetic design [3] and nonlinear equation systems [4]. In such a situation, these problems are generally termed as multimodal optimization problems (MMOPs). Obviously, the traditional implementation of EAs is not capable of locating multiple optima in a single run.

To deal with this issue, niching technique has been developed and embedded into EAs for multimodal optimization [5, 6, 7]. Traditional schemes, such as crowding [8] and fitness sharing [9] generally require specification of certain niching parameters in order to perform well [10]. Recently, several alternative niching schemes [11, 5, 12, 13], which are less sensitive to parameters or require no parameters, have also been proposed. In these methods, the niches are generally used to exploit corresponding subspaces, thus advocating the exploitation aspect of evolutionary search rather than both exploitation and exploration. This could limit their capability of achieving a balanced evolutionary search and therefore restricting the performance to cope with MMOPs. Since different niches have different potential to search the space, it would be desirable to encourage high potential niches for exploitation while low potential niches for exploration, thus properly searching the multimodal space.

On the other hand, niching based EAs are capable of searching multiple peaks in parallel in the solution space. However, they could have difficulty to maintain potential optima recovered during evolution as well as to preserve an appropriate diversity of the population [14], rendering their applications for MMOPs. To alleviate such an issue, archive technique [15, 16, 17, 14] has been proposed and employed to store potential solutions while at the same time to help preserve the population diversity. However, exiting methods of this technique generally require certain external writing and reading rules for implementation. Further, they are typically applied on the population level rather than the niche level. Additionally, in these archive schemes, the population diversity preservation is usually done by detecting the convergence of subpopulation and supplying newly initialized individuals, which may limit the exploitation capability of the evolutionary search.

Apart from locating the regions of optima in the solution space, accurately recovering these optima is also critical. Although niching-based EAs can be suitably used to explore the search space, they are not good at exploiting the space. This could make them difficult to recover the optima with high accuracy. To alleviate this drawback, a few schemes of embedding local searches into EAs, resulting hybrid EAs termed as memetic algorithms [18, 19, 17, 20, 21] have been proposed. Among these schemes, a Gaussian distribution based local search operation recently proposed by Yang et al. [21] appears to be promising. The operation works well for locally improving the solutions, which are close to the optima. However, this is not the case for the ones, which are far away the optima. Thus, restricting its performance to accurately and efficiently identify the optima.

To address the above issues, in this paper, we propose an adaptive memetic differential evolution algorithm with niching competition and supporting archive strategies for MMOPs. The primary contributions are three-fold:

- A niching competition strategy, in which the niches are competitively employed according to their potentials for searching the space, is devised and incorporated to achieve a well-balanced evolutionary search.

- A supporting archive strategy is designed and implemented at the niche level with a dual purpose of helping maintain the potential optima recovered by the niches during evolution and facilitate the evolution of population.

- An adaptive Cauchy-based local operator, which considers the possible locations of optima to implement the local search, is devised and utilized to efficiently and properly improve the niching seeds.

Specifically, the niching competition strategy in the proposed method is devised to competitively employ the niches according to their potentials by encouraging high potential niches for exploitation while low potential niches for exploration, thus properly searching the space to identify multiple optima. This is achieved by firstly measuring the potential of each niche based on its average fitness and diversity. Then, for individuals from the niches of high potential, the recombination is set to be taken place within the same niche with a high possibility, thus encouraging these niches for exploitation. Otherwise, the recombination will have a high possibility to be happened between the niches, therefore encouraging them for exploration. The supporting archive strategy, on the other hand, is designed and implemented at the niching level to help maintain potential optima recovered by the niches during evolution as well as facilitate the evolution of population. In this strategy, the writing and reading of archive is implicitly performed without requiring any external rules. More importantly, the archive individuals are allowed to take part into mutation operation as supporting individuals. Consequently, they can be used to facilitate the evolution of population by diversifying the search at the early stage of evolution while intensifying the search at the later stage of evolution. Finally, a Cauchy-based local operator is introduced to improve the seed solutions of niches according to the possible positions of corresponding optima. This operator will be adaptively applied to improve the seed solutions such that performing the local search in a small step size when they are close to the possible optima, otherwise in a large step size, thus properly and efficiently improving the solutions. The performance of the proposed method has been assessed on benchmark multimodal functions from congress on evolutionary computation 2013 (CEC'2013) as well as on a robot kinematics problem and compared with related methods. The results show that the devised strategies are able to significantly enhance the performance of the proposed method. Also, the results reveal that our method can consistently and accurately locate multiple optima in the solution space and outperform related algorithms.

The remainder of the paper is organized as follows. Following a brief review of related works in Section 2, we describe our proposed method in Section 3. Subsequently, a series of experiments are conducted in Section 4 to evaluate the performance of proposed method. Finally, a summary along with a discussion of future work is given in Section 5.

2. Related works

2.1. Niching technique

Niching technique [6, 22] which tends to form multiple niches in the population, allows EAs to search multiple peaks in parallel, thus locating multiple optima simultaneously. Traditional methods can be broadly classified into two groups. The first group involves schemes, which encourage the mating and/or replacement within similar individuals by adjusting EA operations. Both crowding based methods [8] and restricted tournament selection [23] belong to this group. The second group consists of techniques, which need an explicit distance cutoff to induce the emergence of niches in the search space. Typical examples include sharing based methods [9] and clearing based methods [24]. These traditional methods have been successfully employed in EAs for optimization. However, they usually require certain niching parameters to be set properly. For some niching parameters (e.g., the crowding factor in crowding based niching methods), appropriate values are problem-specific and could vary at different stages of evolution. Configuring them correctly is a difficult issue [12]. While for others (e.g., the niche radius in sharing and clearing based methods), setting them properly requires a *priori* knowledge about the search space, which is typically not available. To alleviate this issue, a promising solution is perhaps to adaptively set the critical parameter during the run of the algorithm and several viable schemes can be found in [25, 26].

Recently, a few niching schemes [11, 5, 12, 13, 21], which are less sensitive to parameters or require no parameters, have been proposed. The most popular approach among these schemes is perhaps the neighborhood-based niching technique. The technique tends to form multiple subpopulations (niches) within a population using the neighborhood information of the individuals. For instance, Gong et al. [5] employed an index-based neighborhood information of individuals to divide a population into equally sized subpopulations. In this method, the individuals evolve by interacting with their neighbors in the same subpopulation. Qu et al. [13] employed a distance-based neighborhood strategy to induce niches in the population. In this method, the offspring is generated by a neighborhood mutation operation within the same niche and each individual therefore evolves toward the optimal position of the corresponding subspace. In [11], a clustering scheme was employed to partition the population, thus obtaining multiple subpopulations (niches) and the offspring are generated using the neighborhood individuals in the same subpopulation. In the above methods, each niche is typically evolving independently (i.e., the recombination is restricted within the same niche) to search the corresponding subspace. The niches in these methods are therefore mainly used to exploit corresponding subspaces, which could limit their capability to deliver a balanced evolutionary search of the space. In [27], a niching-based scheme called dual-strategy is devised to evolve the individuals. In this scheme, the population is first partitioned into subpopulations (i.e., niches) at each generation. After that, each niche is further divided into two equal sets, i.e., a superior set, which contains individuals with high fitness, and an inferior set, which is consisted of individuals with low fitness. During evolution, if an individual subject to mutation is coming from the superior set, then the mutation strategy of DE/lbest/1 is performed on the individual. Otherwise, the DE/current-to-rand/1 mutation strategy will be applied to generate a mutant. This method, therefore, tends to employ different mutation strategies on different individuals within the same niche to implement the evolutionary search. In this work, we propose to competitively employ the niches during the evolutionary search such that encouraging high potential niches for exploitation while low potential niches for exploration, thus properly searching the space to identify multiple optima.

2.2. Archive methods

Archive methods, which can be used to help maintain potential optima as well as preserve population diversity during evolution, have been proposed and employed in multimodal optimization algorithms. For instance, Lacroix et al. [17] designed an archive method by firstly storing the potential optima obtained during evolution into one collection. Base on the information of this collection, an index of regions of the search space, which are considered to be undesirable for further exploration, is then created and stored in another collection. These two sets will be continuously updated during evolution, thus realizing the idea of archiving. Kundu et al. [16] introduced a speciation-based archive strategy to solve dynamic MMOPs. In this method, when a certain change of environment is detected, the population will be firstly partitioned into multiple same-sized subpopulations. Then, half of the individuals in each subpopulation are reinitialized. The newly generated population will finally serve as the initial population to undergo subsequent evolution. Zhang et al. [14] implemented an archiving method by identifying subpopulations (niches) in the population and detecting their convergences. If the subpopulation is deemed to be converged, then all members of the subpopulation are recorded into an archive and re-initialized. In [15], a structure of hierarchical tree is employed to realize the archive. In this method, the nodes located at the top level of tree denote the centers of niches while the nodes beneath each of them represent the individual members of that niche. These nodes will be automatically merged according to a user-specified archive radius. In [27], an archive strategy is designed to store stagnant individuals, which is detected by a stagnation counter. In this method, when a stagnant individual is detected, the stagnant individual along with its neighbors will be reinitialized if their fitness are worse than the stagnant individual. Other archive schemes can be found in [28, 29]. The above archive methods are able to help maintain potential optima and preserve population diversity during evolution. However, they generally require certain external rules for writing and reading the archives. Moreover, these methods are typically applied on the population level rather than the niche level. Additionally, in these archive schemes, the population diversity preservation is usually done by detecting the convergence of subpopulation and supplying newly initialized individuals, which may limit the exploitation capability of evolutionary search. In this paper, a supporting archive strategy, in which the writing and reading of archive is implicitly performed without requiring any external rules, has been proposed and implemented at the niche level to help maintain potential optima. More importantly, the devised archive strategy can facilitate the evolution of population by diversifying the search at the early stage of evolution while intensifying the search at the later stage of evolution, thus properly searching the multimodal space.

2.3. Local searches in memetic algorithms

Traditional EA based multimodal optimization algorithms could fail to accurately recover the optima in solution space, as they are not well suited to exploit the space. To address this problem, local searches have been introduced into these algorithms to improve their exploitation capability, resulting memetic algorithms (MAs). For instance, Vitela et al. [30] integrated a gradient-based search operation into an EA for fine-tuning the individuals and accelerating their convergence to corresponding optima during evolution. In [31], two local search operations (i.e., a simulated annealing-based operator and a chaotic operator) were incorporated into a particle swarm optimization (PSO) algorithm to improve its search capability. In this method, the first operation is employed to improve elite particles around promising regions while the second operation is applied to stagnant particles, whose personal best (*pbest*) cannot be improved further. In [32], a local search based *pbest* mutation operator was devised to improve the exploitation capability of a PSO algorithm for MMOPs. In this method, the mutation operation is used to generate an offspring around the particle's *pbest* by adding a small step. Wang et al. [33] employed two local searches (i.e., random walk with direction exploitation (RWDE) [34] and cognition-based local search operator (CBLS) [35]) to improve the particles in population. In this method, if particles are close to their *pbests*, then RWDE will be applied to guide these particles towards their *pbests*. Otherwise, CBLS is used to improve the particles. In [20], the Solis and Wets' algorithm [36] was incorporated into a PSO algorithm as the local search to improve the newly generated individuals, which possess high fitness values. Sharifi et al. [37] applied the naive directed search [30] for fine-tuning individuals in the population. In [17], Lacroix et al. employed a derandomized evolution strategy with covariance matrix adaptation [38] to improve the best solution in population. Yang et al. [21] devised a Gaussian distribution based local search operation, which tends to fine-tune an individual by performing a sampling search in a narrow space around the individual. In [39], two local searches, namely, Rosenbrock Algorithm (RA) and Nelder Mead Algorithm (NMA), were incorporated into a differential evolution (DE) algorithm and adaptively employed to improve the individuals. In this method, the NMA is set to perform on a randomly selected individual while the RA is applied on the best individual in population. Tirronen et al. [40] integrated three local searches (i.e., the simulated annealing, a stochastic local search and the Hooke-Jeeves algorithm) to assist the evolutionary search of DE. In this method, three local searchers are coordinated via an adaptive scheme to improve the individuals during evolution. In [41], two local searches, which work cooperatively and competitively for improving the individuals, were incorporated into a DE for online and offline control design of permanent-magnet synchronous motor drives. In [22], a two-level local search, which includes a niching-level and an individual-level local search, was devised to improve the accuracy of solution during evolution of DE. Several good reviews in this area can be found in [42, 43] for memetic DE and in [44] for general MA. In this paper, an adaptive Cauchy-based local operator, which considers the possible locations of optima to implement the local search, is devised and utilized to efficiently and properly improve the niching seeds.

3. Proposed algorithm

In this section, we propose a niching competition based memetic DE algorithm with supporting archive and adaptive local search operation for MMOPs. The proposed algorithm starts with an initial population P and archive A as well as an initial DE parameter setting. Then, simply merge all the initial individuals in P and A to form a joint population PA. At each generation, the joint population is firstly partitioned into niches by employing a certain niching method. These niches are then evolved via a niching competition strategy, which is developed to encourage high potential niches for exploitation while low potential niches for exploration. During this process, the devised supporting archive strategy, in which the writing and reading of archive is implicitly implemented without requiring any external rules, is also performed to help maintain the potential optima recovered by the niches as well as facilitate the evolution of population. After that, employing the designed adaptive local operator to improve niche seeds. Finally, a parameter adaptation scheme is employed to update the crossover rate and scaling factor of DE. The evolution will repeat until a maximum number of function evaluations is reached. The outline of the proposed method is shown in Algorithm 1.

Algorithm 1 A niching competition based memetic differential evolution algorithm with supporting archive strategy and adaptive local search for MMOPs.

1: Generate an initial population P as well as an archive A and set initial DE parameter values.

Simply merge all the initial solutions in P and A to form a joint population PA.

Employ a certain niching method to partition PA into niches and calculate their potential values according to equation (1).

Perform the niching competition process (see Section 3.1) to evolve the niches as follows, in which the supporting archive strategy (see Section 3.2) will also be implicitly implemented.

for each niche i **do**

  for each individual p, which is not belonging to archive A, in the niche i **do**

    Generate a random value $rand$ between 0.0 to 1.0.

    Calculate the pr_i using equation (2).

    if $rand$ is less than pr_i **then**

      Apply the DE/rand/1 mutation scheme [32] to produce a mutation vector using the individuals from the niche i.

      Perform the binomial crossover operation [45] to generate a new individual c.

      Pair c with the most similar individual in the niche i and replace it if c has a better fitness.

    else

      Select a niche from the rest niches using the roulette selection strategy based on their affinity values, calculated according to equation (3).

      Randomly choose individuals without replacement from the selected niche.

      Apply the DE/rand/1 mutation scheme to produce a mutation vector using the selected individuals.

      Perform the binomial crossover operation to generate a new individual c.

      Pair c with the most similar individual in PA, and replace it if c has a better fitness.

    end if

20: **end for**

21: **end for**

22: Perform the adaptive Cauchy-based local search operation (see Section 3.3) with a probability of p_s, calculated using equation (7), to improve the seed solutions of niches.

23: Employ the parameter adaptation scheme (see Section 3.4) to update DE parameters.

24: Terminate the evolution when a maximum number of function evaluations is reached. Otherwise go to Step 3.

25: Output the seed solutions of niches.

In the following subsections, we shall give the details of the proposed niching competition, supporting archive, adaptive local search strategies as well as the adopted DE parameter control scheme in the proposed algorithm.

3.1. Niching competition strategy

The neighborhood-based niching methods, which are less sensitive to parameters or require no parameters, have been incorporated into EAs for MMOPs [46, 11, 13]. These methods try to divide the population into multiple niches and each niche is then evolved independently to search its corresponding subspace. By restricting the recombination within the same niche, the niches in these methods are mainly used to exploit their corresponding subspaces, thus limiting their capability to deliver a balanced evolutionary search in terms of exploitation and exploration. Generally, different niches are associated with different subspaces and a niche associated with a promising subspace will have a high potential. Intuitionally, to effectively solve MMOPs, the niches should be employed to ensure a diverse search over the space while allowing promising subspaces to be intensively searched. Following this intuition, here, we intend to competitively employ the niches according to their potentials by encouraging high potential niches for exploitation while low potential niches for exploration, therefore properly searching the space to identify multiple optima.

To realize the above idea, a niching competition strategy has been proposed and it works as follows. Firstly, at the beginning of each generation of evolution, we apply a certain niching method on the population to obtain the niches. Theoretically, the proposed strategy could work with any niching methods, in which the niches can be explicitly identified. Here, a simple while widely used niching method, called speciation cluster niching (SCN), presented in [11] has been employed to partition the population into niches. This niching method forms niches by repeating the following procedure. Firstly, selecting the best individual in the population as the seed, then forming a niche with $w-1$ individuals closest to it (measured by the Euclidean distance in genotype space), and finally removing these w individuals from the population. In this method, the value of w is fixed to be five. After obtaining the niches, the potential of each niche is then evaluated. The potential of a niche depends on its corresponding subspace being searched. A niche associated with a high promising subspace generally has a high average fitness. On the other hand, a high potential niche should also have a high evolvability, which can be measured by its diversity [47]. In this sense, we define the potential of a niche as:

$$PT_i = f_{i,ave} \cdot (f_{i,seed} - f_{i,ave}) \tag{1}$$

where $f_{i,ave}$ and $f_{i,seed}$ are the average fitness and the fitness of seed solution (i.e., the best solution), respectively, of niche i. Here, the term $(f_{i,seed} - f_{i,ave})$ is used to measure the phenotype diversity of the niche. Based on the above equation, a niche with a high average fitness as well as a high phenotype diversity will have a large potential value. It should be noted that the niches in our method are formed by employing the SCN scheme, which produces the niches based on the neighborhood information of individuals (measured by the Euclidean distance in genotype space). Consequently, all of the formed niches will generally have a low genotype diversity. In this sense, a phenotype diversity could be more suitable to be employed here for measuring the diversity of the niche, since a low genotype diversity does not mean a low phenotype diversity.

Based on the calculated potential value of each niche, we subsequently implement the following rule to determine whether the recombination should take place within the niche itself to exploit the corresponding subspace or between the niches to advocate exploration. Let pr_i be the possibility of recombination for an individual from niche i to be happened within the niche, we compute it as:

$$pr_i = \frac{PT_i - PT_{min}}{PT_{max} - PT_{min}} \tag{2}$$

Here, PT_{max} and PT_{min} are the maximum and minimum potential value, respectively, of the niches in the current generation. As a result, individuals from niches with high potential values, the recombination will have a high possibility to be taken place within the same niche, thus encouraging these niches for exploitation. Otherwise, they will have a high possibility to be happened between the niches, therefore encouraging low potential niches for exploration.

The above rule plays its role well to determine whether the niches should be encouraged for exploitation or exploration. The performance of the proposed strategy, however, depends also on how the partner niche should be selected for a niche, which is set to explore the space. The partner niche could be simply chosen by randomly selecting one niche from the rest niches. However, if the paired niches have rather different average fitness, the one with lower average fitness could quickly disappear before its corresponding subspace is being sufficiently searched. While, if the paired niches are far away from each other, the mating could become too destructive and lead to excessive exploration, which is not helpful for searching the multimodal space. Based on the above considerations, we thus encourage the recombination to occur between similar niches in terms of both the average fitness and niche distance. By doing so, we aim to maintain the smoothness of niching competition and to preserve the population diversity at niche level during the process of encouraging low potential niches for exploration, thus properly searching the multimodal space. Specifically, the process of selecting a partner niche for niche i, which is set to explore the space, is implemented as follows. Firstly, we calculate the affinity value, AF_j, for each of the rest niches to the niche i as:

$$AF_j = \frac{d_{max}}{d_{i,j}} \cdot \frac{f_{ave}^{max} - f_{ave}^{min}}{|f_{i,ave} - f_{j,ave}|} \tag{3}$$

Here, $d_{i,j}$ denotes the distance between the seeds of i^{th} and j^{th} niche, d_{max} is the maximum seed distance of all the rest niches to the niche i, $f_{i,ave}$ and $f_{j,ave}$ denote the average fitness of i^{th} and j^{th} niche, respectively, while f_{ave}^{max} and f_{ave}^{min} are the maximum and minimum average fitness among all niches. According to the above equation, a niche, which is near the niche i while having a similar average fitness, will have a high affinity value. Based on the obtained affinity values of all the rest niches, a roulette selection strategy is then employed to select a niche and subsequently choose individuals randomly from the selected niche as the mates for the individual from niche i to generate offspring. The generated offspring will finally pair with the most similar individual in the joint population, and replace it if the offspring shows a higher fitness. The procedure of proposed niching competition strategy is shown in Steps 5-15 of Algorithm 1.

3.2. Supporting archive strategy

Archive technology, which can be used to maintain the potential optima while at the same time preserve the population diversity during evolution, has been widely used in niching based EAs for MMOPs. Existing schemes, however, generally require certain explicit rules for writing and reading the archives. Further, they are typically applied on the population level. Additionally, in these archive schemes, the population diversity preservation is usually done by detecting the convergence of subpopulation and supplying newly initialized individuals. Here, we propose an archive strategy, in which the writing and reading of archive is implicitly performed during evolution, and implement it on the niche level. More importantly, the archive individuals in the proposed strategy are allowed to take part into mutation operation as supporting individuals. Consequently, they can be used to facilitate the evolution of population by diversifying the search at the early stage of evolution while intensifying the search at the later stage of evolution, thus properly searching the multimodal space. The procedure of the proposed strategy can be found in Steps 1-15 of Algorithm 1.

Specifically, the proposed archive strategy, which is termed as supporting archive strategy, works as follow. At the initialization stage, an archive A is randomly generated and merged with the population P to form a joint population PA. The PA is then partitioned into niches for evolution. At each generation during evolution, the individuals from archive A in each niche will go through the evolution process of mutation as supporting individuals (i.e., they are not allowed to generate offspring) as well as replacement. By allowing archive individuals to participate into the mutation operation as supporting individuals, the reading of the archive can thus be implicitly implemented. During the replacement, if an individual belonging to the archive is replaced, then the new individual will be marked as an archive individual, hence implicitly realizing the writing of the archive. By partitioning the joint population PA into niches and evolving them one by one, the above archive strategy is therefore implemented at the niche level rather than population level. Consequently, the archive individuals can be used to help maintain the potential optimum identified by the niches. It should be noted that the proposed archive strategy cannot guarantee the niche seeds will enter the archive. Although such a guarantee can be achieved by scanning the population and marking all the seeds as archive members at each generation, this has little impact on the performance of the proposed strategy and introduces an extra process. This is partially due to, by employing the above archive strategy, most of the seed solutions will be marked as the archive members during evolution, especially at the later stage of evolution.

Particularly, since the archive individuals in the proposed strategy are allowed to take part into the mutation operation as supporting individuals, they can also be viewed as a support population for evolving individuals of the population P. Generally, at the early stage of evolution, the archive population contains individuals with low fitness. By allowing these individuals to participate into mutation as supporting individuals, they can be used to diversify the population P to explore the space. While, at the later stage of evolution, the

Algorithm 2 A Gaussian-based local search scheme

1: For each target solution p to be improved, sample a trial solution tp according to Gaussian(p, 1.0E-4) distribution and truncate it to a preset value range.

2: If tp is better than p, then replace p with tp.

3: If the user-specified number of iterations is not reached, then go to Step 1. Otherwise, output the solution p.

archive population is typically consisted of many individuals with high fitness, which can be served to intensify the search by helping generate highly fitted mutants, thus strengthening the population P to exploit the space. Therefore, apart from serving to store promising individuals, the archive population is able to support the population P to properly search the multimodal space. It should be noted that such a support comes with no extra function evaluation cost, except for calculating the fitness of initial archive individuals.

3.3. Adaptive local search strategy

In this section, an adaptive Cauchy-based local search strategy, which considers the possible positions of optima to implement the local search, is introduced to improve the seed solutions during evolution. Before presenting the proposed strategy, we first briefly describe the Gaussian distribution based local search scheme devised by Yang et al. [21], a work which inspires our strategy. In this scheme, Gaussian distribution with a small standard deviation is utilized for sampling a trial solution tp around the seed solution p. If tp possesses a higher fitness than p, then replaces p with tp. The procedure of the scheme is shown in Algorithm 2. This scheme could work well for locally improving the individuals, which are close to the optima. However, this is not the case for the individuals, which are far away from the optima, thus limiting its performance to efficiently and accurately identify the optima. This is mainly due to the scheme does not consider the possible positions of optima for applying the local search. Further, the adopted Gaussian distribution for sampling has a narrow sampling space.

Algorithm 3 An adaptive Cauchy-based local search strategy

1: For each target solution p to be improved, set the initial sampling center mp as p.
2: **while** a user-specified number of iterations is not reached **do**
3: Sample a solution tp according to Cauchy(mp, 1.0E-4) distribution and truncate it to a preset value range.
4: **if** tp is better than p **then**
5: Replace p with tp and update mp using equation (4).
6: **else**
7: Update mp using equation (5).
8: **end if**
9: **end while**
10: Output the solution p.

To address the above issue, here we devise an adaptive local search strategy aiming at efficiently and properly improving the seed solution regardless of its distance to the optimum. This is achieved by dynamically updating the sampling center based on the possible location of optimum as well as by employing a Cauchy distribution based sampling, which has a much wider sample space compared with Gaussian distribution. Specifically, to improve a certain seed solution p, the proposed strategy works as follows. Firstly, setting a sampling center mp to be the same as the solution p. Then, sampling a trial solution, denoted as tp, based on a Cauchy distribution. If tp has a better fitness than p, then it is reasonable to assume that the corresponding optimum may locate at the direction of p to tp. In this sense, we replace p as tp and update the sampling center mp as:

$$mp = (1 + \lambda) \cdot tp - \lambda \cdot p \tag{4}$$

Otherwise, if tp is worse than p, the corresponding optimum is highly possible locating at the direction of tp to p. The mp is thus updated as:

$$mp = (1 + \lambda) \cdot p - \lambda \cdot tp \tag{5}$$

where λ is a parameter, which is used to control the step size of updating mp. Here, the solutions p, tp and mp are encoded using a d-dimensional real vector, such as $p = \{p_1, p_2, \ldots, p_d\}$.

Obviously, the value of λ should be set appropriately in order for the proposed local search to perform well. A too small value of which will compromise the efficiency of local search, while a too large value will lead to a dramatic change of the seed solution and may miss the optimum. Intuitionally, the value should be set according to the possible distance of the seed solution to its corresponding optimum. A large value should be used for the seeds, which are far away from the possible optima to improve the efficiency of local search. Otherwise, a small value should be used to avoid missing the possible optima. Generally, the seeds with higher fitness values could be closer to their corresponding optima. In this sense, the following procedure has been introduced to adaptively control the value of λ. Firstly, all the seeds are sorted according to their fitness and ranked from 1 to S. For each seed i, the λ value is then calculated as:

$$\lambda = 0.5 + 0.5 \cdot \frac{Rank(i) - 1}{S} \tag{6}$$

According to the above equation, a large λ value will be used for seeds, which could be far away from the possible optima. Otherwise, a small value of λ will be adopted. In other words, the local search will be adaptively employed to improve the seed solutions such that performing the local search in a large step size when they are far away to the possible optima, otherwise in a small step size. The above procedure will be repeatedly employed to improve the seed solution and the value of λ will be dynamically calculated at each iteration. Consequently, by employing the above strategy, for a seed, which is far away from the possible optimum, it will quickly approach to a position near the possible optimum. While, for a seed, which is near the possible optimum, a fine-tuning will be performed to properly locate the optimum. It should be noted that in case the local landscape is in U shape and the seed solution is located at a position near the bottom of the U shape, the local search will tend to fine-tune the seed since, according to equation (6), a small value of λ will generally be adopted. This could affect the efficiency of local search to improve the seed. However, due to the local search is iteratively employed and population of DE is evolved generationally, the seed solution will gradually approach to the optimal position of U shape landscape along with the evolution. It should also be noted that rather than applying the proposed local search to improve the seed solutions at every generation, it will be evoked with a probability of p_s at each generation. The p_s is computed as:

$$p_s = \frac{NCFE}{MNFE} \tag{7}$$

where NCFE and MNFE denote the number of function evaluation consumed so far and the maximum number of function evaluation, respectively. Consequently, the possibility of employing the local search will gradually increase along with the evolution. This will allow the population to explore the space at the early stage of evolution while intensify the search towards the end of evolution. The procedure of the proposed local search strategy is shown in Algorithm 3.

3.4. DE parameter adaptation

To set the scaling factor F and crossover rate CR in DE, a parameter adaptation scheme presented in [14] has been employed. The scheme works by maintaining a historical memory with L entries for scaling factor F as well as crossover rate CR, denoted as M_F and M_{CR}, which are initialized to be 0.5. At each generation, when a solution p is subject to recombination, an index I_p is first randomly chosen between 1 to L. The values of F_p and CR_p for this individual are then computed to be:

$$\begin{cases} F_p = Cauchy(M_{F,I_p}, 0.1) \\ CR_p = Gaussian(M_{CR,I_p}, 0.1) \end{cases} \tag{8}$$

After performing recombination operations, if the generated offspring has a better fitness than its paired solution, then F_p and CR_p will be inserted into

S_F and S_{CR}, respectively. When all individuals in the population have been processed according to the above procedure at each generation, M_F and M_{CR} will be updated as:

$$\begin{cases} M_{F,k} = mean_{WL}(S_F) \\ M_{CR,k} = mean_{WA}(S_{CR}) \end{cases} \tag{9}$$

where k (with an initial value of 1) denotes the position in M_{CR} and M_F. The value of k will increase by 1 after each updating of M_F and M_{CR}, and reset to 1 when it reaches a value larger than L. The weighted Lehmer mean, $mean_{WL}(S_F)$, and weighed mean, $mean_{WA}(S_{CR})$, in the above equation are defined as:

$$\begin{cases} mean_{WL}(S_F) = \frac{\sum_{p=1}^{|S_F|} w_p \cdot S_{F,p}^2}{\sum_{p=1}^{|S_F|} w_p \cdot S_{F,p}} \\ mean_{WA}(S_{CR}) = \sum_{p=1}^{|S_{CR}|} w_p \cdot S_{CR,p} \end{cases} \tag{10}$$

Here, w_p is calculated as:

$$w_p = \frac{\Delta f_p}{\sum_{i=1}^{|S_{CR}|} \Delta f_i} \tag{11}$$

where Δf_p represents the fitness improvement of the offspring p compared with its paired solution.

4. Experiments

A series of experiments have been carried out to access the significance of devised strategies as well as to compare the proposed method with state-of-the-art multimodal optimization algorithms. All methods are implemented using C++ and tested on a workstation with an Intel (R) $Core^{TM}$ i7-3630QM CPU at 2.40GHz and 8 GB RAM running $Windows^{TM}$ 10 operation system. Unless otherwise stated, 100 independent trials are performed for each method and the average results are reported.

4.1. Experimental data and settings

The benchmark dataset from CEC'2013 [48] as well as a robot kinematics problem (RKP) [49] have been used for experiments. The CEC'2013 dataset contains 20 multimodal functions with various characteristics and different levels of difficulty to be solved. These functions are generally designed or proposed to contain many equal peaks (i.e., many global optima) and have been widely used to test the performance of multimodal optimization methods by evaluating their capability to identify the global optima. The properties of these functions are shown in Table S1 in the supplement document. The robot kinematics problem taken from [49] is cast as a system of nonlinear equations. The problem has multiple roots and each root could be equally important. The task of solving this problem is thus to identify all the roots. A detailed description of this problem is shown in the supplement document.

To evaluate the performance, two commonly used indexes, i.e., the peak ration (PR) and success rate (SR), have been adopted. Given a maximum number of function evaluation (MNFE) and a user-specified level of accuracy, PR measures the average percentage of optima found in all known optima over multiple trials while SR counts the rate of successful trials, in which all known optima can be identified. Specifically, the PR and SR are calculated as:

$$PR = \frac{\sum_{i=1}^{T} NPK_i}{NPK \cdot T} \tag{12}$$

$$SR = \frac{NSR}{T} \tag{13}$$

where T denotes the total number of trials, NPK_i is the number of optima found in the i^{th} trial, NPK is the total number of optima and NSR represents the number of successful trials.

Table 1: Parameter settings.

Problem	Population Size	MNFE
F1-F5	80	5.0E+4
F6	100	2.0E+5
F7	300	2.0E+5
F8-F9	300	4.0E+5
F10	100	2.0E+5
F11-F13	200	2.0E+5
F14-F20	200	4.0E+5
RKP	100	4.0E+5

In experiments, we adopt five levels of accuracy (ϵ=1.0E-1, 1.0E-2, 1.0E-3, 1.0E-4, 1.0E-5) to evaluate the methods. On robot kinematics problem, three additional levels of accuracy, i.e., ϵ=1.0E-6, 1.0E-7, 1.0E-8 have also been used for evaluation. To make fair comparisons, the same population size and MNFE value are used for all algorithms on each problem. Table 1 shows the settings of population size and MNFE on the test problems. The population sizes and MNFEs are set according to the complexity degrees of the problems. For a problem with a large number of optima, it will be allocated with a large population size and MNFE. Such a setting for the benchmark functions is consistent with previous studies [46, 11, 13]. The two parameters in the proposed method, i.e., the archive size in the supporting archive strategy and the number of iterations of performing the adaptive local search to improve the seeds, are experimentally determined based on the problems. To set the archive size, generally, we found that for the problems with complex search spaces and a large number of optima, a large archive size will lead to a better result. This is due to a larger size of archive will help the population to perform a more diverse search especially at the early stage of evolution, which is beneficial to search a complex search space and locate a large number of optima. This, however, is not the case for the problems with a relatively smaller number of optima. Rather, in this case, a too large archive size will lead to an excessive exploration, thus could significantly reduce the efficiency of convergence of the population. To simplify the setting of archive size as well as to deliver a balanced performance over various problems, we set it to be the same as the population size. Certainly, this parameter could be more effectively set to achieve an even better performance. For the number of iterations, it is set to be 3. Generally, we found that either a smaller or larger value of this parameter will hinder the performance of the method. This is due to a smaller number of iterations will restrain the efficiency of improving seed solutions, resulting in a slow convergence of the niches. While, a larger number of iterations could lead the niches to convergence prematurely before the corresponding subspaces being sufficiently searched.

Table 2: Comparing results delivered by the NSAMA and its three variants in term of PR with the best PR values bolded.

	NSAMA	**NSAMA_1**	**NSAMA_2**	**NSAMA_3**	**NSAMA**	**NSAMA_1**	**NSAMA_2**	**NSAMA_3**	**NSAMA**	**NSAMA_1**	**NSAMA_2**	**NSAMA_3**	**NSAMA**	**NSAMA_1**	**NSAMA_2**	**NSAMA_3**
ε	F1				F6				F11				F16			
1.0E-1	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.999	0.991	0.852	**1.000**	**1.000**	0.998	0.998	0.869	**0.922**	0.745	0.773
1.0E-2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.999	0.991	0.849	**1.000**	**1.000**	0.997	0.993	**0.667**	**0.667**	0.665	0.657
1.0E-3	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.998	0.991	0.847	**0.998**	0.995	0.993	0.990	**0.667**	**0.667**	0.665	0.655
1.0E-4	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.996	0.991	0.847	**0.997**	0.993	0.993	0.990	**0.667**	0.665	0.660	0.652
1.0E-5	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.996	0.989	0.844	**0.993**	**0.993**	**0.993**	0.988	**0.667**	0.665	0.657	0.652
ε	F2				F7				F12				F17			
1.0E-1	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.999	0.841	0.572	**1.000**	0.999	0.988	0.990	**0.750**	0.663	0.585	0.634
1.0E-2	**1.000**	**1.000**	**1.000**	**1.000**	**0.942**	0.931	0.841	0.571	**1.000**	0.998	0.981	0.981	**0.709**	0.524	0.493	0.483
1.0E-3	**1.000**	**1.000**	**1.000**	**1.000**	**0.942**	0.928	0.841	0.570	**1.000**	0.994	0.966	0.976	**0.704**	0.471	0.435	0.421
1.0E-4	**1.000**	**1.000**	**1.000**	**1.000**	**0.942**	0.922	0.839	0.566	**0.999**	0.991	0.954	0.960	**0.650**	0.425	0.400	0.383
1.0E-5	**1.000**	**1.000**	**1.000**	**1.000**	**0.942**	0.910	0.831	0.560	**0.995**	0.990	0.950	0.953	**0.521**	0.389	0.371	0.356
ε	F3				F8				F13				F18			
1.0E-1	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.999	0.703	0.697	0.723	0.733	**0.755**	**0.855**	0.792	0.485	0.477
1.0E-2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.999	0.703	0.667	0.675	0.698	**0.707**	**0.647**	0.408	0.393	0.405
1.0E-3	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.996	0.999	0.703	0.667	0.670	**0.682**	0.680	**0.645**	0.388	0.380	0.385
1.0E-4	**1.000**	**1.000**	**1.000**	**1.000**	0.997	0.986	**0.999**	0.703	0.667	0.668	0.677	**0.678**	**0.642**	0.373	0.373	0.370
1.0E-5	**1.000**	**1.000**	**1.000**	**1.000**	0.857	0.959	**0.999**	0.703	0.667	0.667	**0.675**	0.672	**0.598**	0.365	0.355	0.362
ε	F4				F9				F14				F19			
1.0E-1	**1.000**	**1.000**	**1.000**	**1.000**	**0.620**	0.615	0.436	0.205	0.740	**0.805**	0.691	0.707	**0.493**	0.371	0.315	0.358
1.0E-2	**1.000**	**1.000**	**1.000**	**1.000**	**0.588**	0.581	0.436	0.205	**0.667**	**0.667**	**0.667**	**0.667**	**0.466**	0.315	0.285	0.309
1.0E-3	**1.000**	**1.000**	**1.000**	**1.000**	**0.587**	0.581	0.436	0.205	**0.667**	**0.667**	**0.667**	**0.667**	**0.465**	0.296	0.278	0.298
1.0E-4	**1.000**	**1.000**	**1.000**	**1.000**	**0.587**	0.577	0.436	0.205	**0.667**	**0.667**	**0.667**	**0.667**	**0.464**	0.273	0.268	0.285
1.0E-5	**1.000**	**1.000**	**1.000**	**1.000**	**0.586**	0.572	0.435	0.205	**0.667**	**0.667**	**0.667**	**0.667**	**0.438**	0.256	0.256	0.274
ε	F5				F10				F15				F20			
1.0E-1	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.987	0.806	**0.830**	0.744	0.758	0.269	0.269	**0.276**	0.273
1.0E-2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.998	**1.000**	0.987	**0.750**	0.746	0.726	0.740	**0.251**	0.245	0.248	0.249
1.0E-3	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.998	**1.000**	0.987	**0.750**	0.746	0.716	0.730	**0.250**	0.244	0.246	0.249
1.0E-4	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.998	**1.000**	0.987	**0.749**	0.741	0.706	0.720	0.246	0.243	0.246	**0.249**
1.0E-5	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.998	**1.000**	0.987	0.739	**0.740**	0.699	0.715	0.246	0.239	0.245	**0.249**

4.2. Exploring the proposed method

First, we evaluate the significance of proposed niching competition (NC), supporting archive (SA) and adaptive local search (ALS) strategies in the proposed algorithm. For this purpose, we carry out experiments to compare the proposed algorithm (denoted as NSAMA) with its three variants: NSAMA without ALS (NSAMA_1), NSAMA without ALS and SA (NSAMA_2) and NSAMA without all the above three strategies (NSAMA_3) on the benchmark functions. The above four algorithms are evaluated utilizing the same parameter settings. Table 2 reports the results in term of PR of the four methods. The values of different peaks identified by a typical run of NSAMA at an accuracy level of ϵ=1.0E-5 have also been shown in Table S1 in the supplement document.

As can be found from the results of Table 2, the three proposed strategies are able to significantly enhance the algorithm's performance. Specifically, incorporated with the NC strategy, NSAMA_2 can generally locate more optima than NSAMA_3 on the multimodal functions. Particularly, on functions F6-F11, the

Table 3: Comparing results delivered by our proposed method and seven related methods at accuracy level ϵ=1.0e-1.

F	CDE		LISDE		LICDE		NSDE		NCDE		SCSDE		SCCDE		NSAMA	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F1	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F2	**1.000**	1.000	0.252	0.000	**1.000**	1.000	0.396	0.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F3	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F4	**1.000**	1.000	0.250	0.000	**1.000**	1.000	0.250	0.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F5	**1.000**	1.000	0.500	0.000	**1.000**	1.000	0.500	0.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F6	**1.000**	1.000	0.056	0.000	0.984	0.780	0.056	0.000	0.969	0.660	0.444	0.000	**1.000**	1.000	**1.000**	1.000
F7	0.997	0.940	0.030	0.000	0.935	0.130	0.036	0.000	0.933	0.170	0.452	0.000	0.881	0.030	**1.000**	0.980
F8	0.008	0.000	0.012	0.000	0.838	0.000	0.012	0.000	0.999	0.950	0.187	0.000	0.999	0.900	**1.000**	0.970
F9	0.503	0.000	0.005	0.000	0.483	0.000	0.005	0.000	0.451	0.000	0.121	0.000	0.462	0.000	**0.620**	0.000
F10	**1.000**	1.000	0.087	0.000	**1.000**	1.000	0.083	0.000	**1.000**	1.000	0.994	0.930	**1.000**	1.000	**1.000**	1.000
F11	0.993	0.960	0.173	0.000	**1.000**	1.000	0.168	0.000	0.778	0.160	0.993	0.960	0.987	0.920	**1.000**	1.000
F12	0.159	0.000	0.125	0.000	0.810	0.120	0.125	0.000	0.706	0.020	0.805	0.140	0.968	0.790	**1.000**	1.000
F13	**0.932**	0.660	0.168	0.000	0.708	0.040	0.183	0.000	0.693	0.000	0.847	0.250	0.678	0.010	0.697	0.000
F14	**0.857**	0.420	0.167	0.000	0.722	0.050	0.173	0.000	0.750	0.140	0.673	0.000	0.752	0.080	0.740	0.140
F15	**0.970**	0.870	0.125	0.000	0.563	0.010	0.125	0.000	0.491	0.000	0.523	0.000	0.600	0.000	0.806	0.170
F16	**0.957**	0.920	0.165	0.000	0.928	0.670	0.157	0.000	0.905	0.570	0.605	0.000	0.940	0.730	0.869	0.510
F17	0.110	0.010	0.078	0.000	0.489	0.000	0.086	0.000	0.308	0.000	0.335	0.000	0.460	0.000	**0.750**	0.070
F18	0.648	0.260	0.148	0.000	0.872	0.650	0.087	0.000	**0.960**	0.850	0.300	0.000	0.950	0.790	0.855	0.460
F19	0.000	0.000	0.015	0.000	0.190	0.000	0.013	0.000	0.313	0.030	0.190	0.000	0.483	0.010	**0.493**	0.000
F20	0.129	0.080	0.074	0.000	0.129	0.000	0.006	0.000	0.281	0.000	0.173	0.000	**0.528**	0.010	0.269	0.000
bprs	11		2		7		2		7		5		8		14	

NSAMA_2 achieves much better PR values than NSAMA_3. Consequently, by encouraging high potential niches for exploitation while low potential niches for exploration, the NC strategy can be used to properly search the space to identify multiple optima. Looking at NSAMA_1 and NSAMA_2, the results show that the SA strategy can benefit the NSAMA_1 especially on functions F6-F7, F9, F11-F12 and F15-F19. This is due to the proposed archive strategy helps appropriately maintain the potential optima recovered during evolution as well as facilitate the evolution of population. By examining NSAMA and NSAMA_-1, we can find that the local search helps to accurately identify the optima on all functions except F14-F16. Based on the results, it is clear that NC, SA and ALS strategies could greatly improve the algorithm's search capability for multimodal space, thus effectively locating the optima.

4.3. Comparing with related algorithms

Then, we access the performance of the proposed algorithm by comparing it with recently proposed multimodal optimization methods including crowding-based DE (CDE) [50], locally informative speciation-based DE (LISDE) [46], locally informative crowding-based DE (LICDE) [46], neighborhood based species DE (NSDE) [13], neighborhood based crowding DE (NCDE) [13], cluster-based

Table 4: Comparing results delivered by our proposed method and seven related methods at accuracy level ϵ=1.0e-2.

F	CDE		LISDE		LICDE		NSDE		NCDE		SCSDE		SCCDE		NSAMA	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F1	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F2	**1.000**	1.000	0.252	0.000	**1.000**	1.000	0.396	0.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F3	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F4	**1.000**	1.000	0.250	0.000	**1.000**	1.000	0.250	0.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F5	**1.000**	1.000	0.500	0.000	**1.000**	1.000	0.500	0.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F6	0.998	0.960	0.056	0.000	0.879	0.140	0.056	0.000	0.859	0.120	0.444	0.000	**1.000**	1.000	**1.000**	1.000
F7	0.887	0.020	0.030	0.000	0.890	0.000	0.036	0.000	0.869	0.000	0.452	0.000	0.872	0.020	**0.942**	0.140
F8	0.000	0.000	0.012	0.000	0.620	0.000	0.012	0.000	0.999	0.950	0.187	0.000	0.999	0.900	**1.000**	0.970
F9	0.475	0.000	0.005	0.000	0.482	0.000	0.005	0.000	0.449	0.000	0.121	0.000	0.451	0.000	**0.588**	0.000
F10	**1.000**	1.000	0.087	0.000	**1.000**	1.000	0.083	0.000	**1.000**	1.000	0.994	0.930	**1.000**	1.000	**1.000**	1.000
F11	0.667	0.000	0.173	0.000	**1.000**	1.000	0.168	0.000	0.678	0.000	0.993	0.960	0.953	0.740	**1.000**	1.000
F12	0.014	0.000	0.125	0.000	0.690	0.000	0.125	0.000	0.401	0.000	0.805	0.140	0.779	0.040	**1.000**	1.000
F13	0.588	0.000	0.168	0.000	0.667	0.000	0.183	0.000	0.667	0.000	**0.847**	0.250	0.667	0.000	0.667	0.000
F14	0.662	0.000	0.167	0.000	0.667	0.000	0.173	0.000	0.667	0.000	**0.668**	0.000	0.667	0.000	0.667	0.000
F15	0.231	0.000	0.125	0.000	0.444	0.000	0.125	0.000	0.333	0.000	0.521	0.000	0.471	0.000	**0.750**	0.000
F16	0.235	0.000	0.165	0.000	**0.667**	0.000	0.155	0.000	0.665	0.000	0.587	0.000	**0.667**	0.000	**0.667**	0.000
F17	0.000	0.000	0.078	0.000	0.301	0.000	0.085	0.000	0.245	0.000	0.319	0.000	0.260	0.000	**0.709**	0.000
F18	0.245	0.000	0.148	0.000	0.277	0.000	0.085	0.000	0.337	0.000	0.287	0.000	0.428	0.000	**0.647**	0.000
F19	0.000	0.000	0.015	0.000	0.141	0.000	0.008	0.000	0.169	0.000	0.185	0.000	0.320	0.000	**0.466**	0.000
F20	0.000	0.000	0.074	0.000	0.129	0.000	0.004	0.000	0.231	0.000	0.161	0.000	0.245	0.000	**0.251**	0.000
bprs	6		2		8		2		6		7		8		18	

crowding DE with self-adaptive strategy (SCSDE) [11] and cluster-based species DE with self-adaptive strategy (SCCDE) [11]. In CDE [50], a standard DE incorporated with a crowding strategy is devised to cope with MMOPs. The LISDE and LICDE [46] try to deal with MMOPs with a species- and crowding-based DE, respectively, along with a mutation strategy based on local information sharing. In NSDE [13], a species-based DE with a neighborhood mutation is employed for MMOPs. The NCDE [13] is a crowding-based DE along with a neighborhood mutation scheme. While, in SCSDE and SCCDE [11], a cluster-based self-adaptive DE embedded with a species- and crowding-based niching scheme, respectively, are proposed for multimodal optimization. To make a meaningful comparison, the same population size and MNFE value (see Table 1) are used for all experiments on each problem. Other parameters of the seven methods to be compared are specified or chosen in accordance with their original settings with the best performance.

Tables 3-7 show the comparison results of the methods on the benchmark functions at different accuracy levels. The last row "*bprs*" in each table represents the number of functions where the best PR values are obtained by the corresponding algorithm. The results show that our method can consistently

Table 5: Comparing results delivered by our proposed method and seven related methods at accuracy level ϵ=1.0e-3.

F	CDE		LISDE		LICDE		NSDE		NCDE		SCSDE		SCCDE		NSAMA	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F1	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F2	**1.000**	1.000	0.252	0.000	**1.000**	1.000	0.396	0.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F3	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F4	**1.000**	1.000	0.250	0.000	**1.000**	1.000	0.250	0.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F5	**1.000**	1.000	0.500	0.000	**1.000**	1.000	0.500	0.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F6	0.862	0.140	0.056	0.000	0.669	0.000	0.056	0.000	0.686	0.000	0.444	0.000	0.998	0.970	**1.000**	1.000
F7	0.843	0.000	0.030	0.000	0.889	0.000	0.036	0.000	0.866	0.000	0.452	0.000	0.872	0.020	**0.942**	0.140
F8	0.000	0.000	0.012	0.000	0.428	0.000	0.012	0.000	0.999	0.950	0.187	0.000	0.999	0.900	**1.000**	0.930
F9	0.473	0.000	0.005	0.000	0.482	0.000	0.005	0.000	0.448	0.000	0.121	0.000	0.451	0.000	**0.587**	0.000
F10	**1.000**	1.000	0.087	0.000	**1.000**	1.000	0.083	0.000	0.998	0.980	0.994	0.930	**1.000**	1.000	**1.000**	1.000
F11	0.660	0.000	0.173	0.000	0.997	0.980	0.168	0.000	0.672	0.000	0.990	0.940	0.882	0.460	**0.998**	0.990
F12	0.001	0.000	0.125	0.000	0.540	0.000	0.125	0.000	0.213	0.000	0.805	0.140	0.633	0.000	**1.000**	1.000
F13	0.297	0.000	0.168	0.000	0.667	0.000	0.183	0.000	0.662	0.000	**0.847**	0.250	0.667	0.000	0.667	0.000
F14	0.508	0.000	0.167	0.000	**0.667**	0.000	0.173	0.000	**0.667**	0.000	**0.667**	0.000	**0.667**	0.000	**0.667**	0.000
F15	0.088	0.000	0.125	0.000	0.411	0.000	0.125	0.000	0.301	0.000	0.520	0.000	0.380	0.000	**0.750**	0.000
F16	0.017	0.000	0.165	0.000	**0.667**	0.000	0.153	0.000	0.663	0.000	0.582	0.000	**0.667**	0.000	**0.667**	0.000
F17	0.000	0.000	0.078	0.000	0.274	0.000	0.085	0.000	0.245	0.000	0.308	0.000	0.253	0.000	**0.704**	0.000
F18	0.185	0.000	0.148	0.000	0.260	0.000	0.085	0.000	0.335	0.000	0.285	0.000	0.405	0.000	**0.645**	0.000
F19	0.000	0.000	0.015	0.000	0.139	0.000	0.008	0.000	0.124	0.000	0.180	0.000	0.246	0.000	**0.465**	0.000
F20	0.000	0.000	0.074	0.000	0.129	0.000	0.000	0.000	0.231	0.000	0.158	0.000	0.235	0.000	**0.250**	0.000
bprs	6		2		8		2		6		7		8		19	

outperform the seven methods to be compared. For example, at the accuracy level of ϵ=1.0E-1, our method achieves a *bprs* value of 14. While, the CDE, LISDE, LICDE, NSDE, NCDE, SCSDE and SCCDE give 11, 2, 7, 2, 7, 5 and 8, respectively. More importantly, the results show that our method could be more effective than the seven methods on higher levels of accuracy. For instance, the *bprs* values of CDE, LISDE, LICDE, NSDE, NCDE, SCSDE and SCCDE at the accuracy level of ϵ=1.0E-5 turn out to be 5, 2, 6, 2, 7, 7 and 9, respectively. By contrast, our method gives 18. By examining the results across all levels of accuracy, it can be seen that, on functions of F1 to F6 and F10, NSAMA can recover all known optima. On F7, F9, F11-F12, F17 and F19, NSAMA performs significantly better than all the methods to be compared. While, on functions F15, F16, F18 and F20, our method achieves the best performance except for the accuracy level of ϵ=1.0E-1.

Table 8 shows the PR performance of the methods on robot kinematics problem with eight different accuracy levels. The roots identified by a typical run of NSAMA at an accuracy level of ϵ=1.0E-4 have also been shown in Table S3 in the supplement document. The results in Table 8 show that NSAMA could significantly outperform the related methods to be compared across all levels

Table 6: Comparing results delivered by our proposed method and seven related methods at accuracy level ϵ=1.0e-4.

F	CDE		LISDE		LICDE		NSDE		NCDE		SCSDE		SCCDE		NSAMA	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F1	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F2	**1.000**	1.000	0.252	0.000	**1.000**	1.000	0.396	0.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F3	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F4	**1.000**	1.000	0.250	0.000	**1.000**	1.000	0.250	0.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F5	**1.000**	1.000	0.500	0.000	**1.000**	1.000	0.500	0.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F6	0.411	0.000	0.056	0.000	0.474	0.000	0.056	0.000	0.510	0.000	0.444	0.000	0.998	0.970	**1.000**	1.000
F7	0.606	0.000	0.030	0.000	0.880	0.000	0.036	0.000	0.860	0.000	0.452	0.000	0.872	0.020	**0.942**	0.140
F8	0.000	0.000	0.012	0.000	0.282	0.000	0.012	0.000	**0.999**	0.950	0.187	0.000	**0.999**	0.900	0.997	0.790
F9	0.404	0.000	0.005	0.000	0.472	0.000	0.005	0.000	0.447	0.000	0.121	0.000	0.451	0.000	**0.587**	0.000
F10	**1.000**	1.000	0.087	0.000	**1.000**	1.000	0.083	0.000	0.996	0.950	0.994	0.930	**1.000**	1.000	**1.000**	1.000
F11	0.330	0.000	0.173	0.000	0.990	0.940	0.168	0.000	0.670	0.000	0.990	0.940	0.852	0.340	**0.997**	0.980
F12	0.000	0.000	0.125	0.000	0.371	0.000	0.125	0.000	0.133	0.000	0.803	0.140	0.546	0.000	**0.999**	0.990
F13	0.067	0.000	0.168	0.000	0.667	0.000	0.183	0.000	0.648	0.000	**0.847**	0.250	0.667	0.000	0.667	0.000
F14	0.183	0.000	0.167	0.000	**0.667**	0.000	0.173	0.000	**0.667**	0.000	**0.667**	0.000	**0.667**	0.000	**0.667**	0.000
F15	0.011	0.000	0.125	0.000	0.393	0.000	0.125	0.000	0.283	0.000	0.518	0.000	0.360	0.000	**0.749**	0.000
F16	0.000	0.000	0.165	0.000	**0.667**	0.000	0.153	0.000	0.660	0.000	0.573	0.000	**0.667**	0.000	**0.667**	0.000
F17	0.000	0.000	0.078	0.000	0.265	0.000	0.084	0.000	0.244	0.000	0.299	0.000	0.250	0.000	**0.650**	0.000
F18	0.168	0.000	0.148	0.000	0.250	0.000	0.085	0.000	0.328	0.000	0.283	0.000	0.382	0.000	**0.642**	0.000
F19	0.000	0.000	0.015	0.000	0.136	0.000	0.006	0.000	0.094	0.000	0.175	0.000	0.180	0.000	**0.464**	0.000
F20	0.000	0.000	0.074	0.000	0.129	0.000	0.000	0.000	0.231	0.000	0.153	0.000	0.190	0.000	**0.246**	0.000
bprs	6		2		8		2		7		7		9		18	

of accuracy. Specifically, the results show that the CDE, LISDE, LICDE and NSDE may even fail to identify the optima at lower accuracy levels. By comparison, the NCDE, SCSDE and SCCDE perform reasonably well at lower levels of accuracy. While, along with the increasing of accuracy level, the performance of NCDE and SCSDE could drop dramatically. For instance, at an accuracy level of ϵ=1.0E-4, the NCDE and SCSDE give PR values of 0.136 and 0.050, respectively, which means none of these methods could locate more than two optima in a typical run. By contrast, NSAMA achieves a PR value of 0.830. Among the methods to be compared, the SCCDE turns out to have the best performance. However, its performance would also significantly decline at an even higher accuracy level. For instance, at an accuracy level of ϵ=1.0E-7, the SCCDE delivers a PR value of 0.288, while NSAMA gives 0.355. The results thus further confirm that our method is a viable approach for MMOPs. It should be noted that the performance of EA based multimodal optimization methods depends on the specified level of accuracy for locating the optima. By examining the PR and SR results of each method across different levels of accuracy, it can be found that the performance of all methods generally tends to degrade along with the increasing of accuracy level. This is due to along with the in-

Table 7: Comparing results delivered by our proposed method and seven related methods at accuracy level ϵ=1.0e-5.

F	CDE		LISDE		LICDE		NSDE		NCDE		SCSDE		SCCDE		NSAMA	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F1	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F2	**1.000**	1.000	0.252	0.000	**1.000**	1.000	0.396	0.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F3	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F4	0.815	0.260	0.250	0.000	**1.000**	1.000	0.250	0.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F5	**1.000**	1.000	0.500	0.000	**1.000**	1.000	0.500	0.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F6	0.099	0.000	0.056	0.000	0.313	0.000	0.056	0.000	0.341	0.000	0.444	0.000	0.994	0.920	**1.000**	1.000
F7	0.266	0.000	0.030	0.000	0.826	0.000	0.036	0.000	0.851	0.000	0.452	0.000	0.872	0.020	**0.942**	0.140
F8	0.000	0.000	0.012	0.000	0.176	0.000	0.012	0.000	**0.999**	0.950	0.187	0.000	**0.999**	0.900	0.857	0.000
F9	0.135	0.000	0.005	0.000	0.424	0.000	0.005	0.000	0.445	0.000	0.121	0.000	0.451	0.000	**0.586**	0.000
F10	**1.000**	1.000	0.087	0.000	0.999	0.990	0.083	0.000	0.993	0.920	0.994	0.930	**1.000**	1.000	**1.000**	1.000
F11	0.058	0.000	0.173	0.000	0.973	0.840	0.168	0.000	0.668	0.000	0.990	0.940	0.835	0.270	**0.993**	0.960
F12	0.000	0.000	0.125	0.000	0.214	0.000	0.125	0.000	0.098	0.000	0.803	0.140	0.466	0.000	**0.995**	0.960
F13	0.008	0.000	0.168	0.000	0.667	0.000	0.183	0.000	0.628	0.000	**0.847**	0.250	0.665	0.000	0.667	0.000
F14	0.018	0.000	0.167	0.000	**0.667**	0.000	0.173	0.000	**0.667**	0.000	**0.667**	0.000	**0.667**	0.000	**0.667**	0.000
F15	0.000	0.000	0.125	0.000	0.379	0.000	0.125	0.000	0.275	0.000	0.518	0.000	0.358	0.000	**0.739**	0.000
F16	0.000	0.000	0.165	0.000	0.662	0.000	0.153	0.000	0.660	0.000	0.568	0.000	**0.667**	0.000	**0.667**	0.000
F17	0.000	0.000	0.078	0.000	0.256	0.000	0.084	0.000	0.244	0.000	0.295	0.000	0.250	0.000	**0.521**	0.000
F18	0.163	0.000	0.148	0.000	0.245	0.000	0.080	0.000	0.327	0.000	0.275	0.000	0.363	0.000	**0.598**	0.000
F19	0.000	0.000	0.015	0.000	0.135	0.000	0.005	0.000	0.080	0.000	0.170	0.000	0.131	0.000	**0.438**	0.000
F20	0.000	0.000	0.074	0.000	0.129	0.000	0.000	0.000	0.231	0.000	0.153	0.000	0.139	0.000	**0.246**	0.000
bprs	5		2		6		2		7		7		9		18	

Table 8: Comparing PR values delivered by our proposed method and seven related methods on the robot kinematics problem at various accuracy levels.

Accuracy Level	CDE	LISDE	LICDE	NSDE	NCDE	SCSDE	SCCDE	NSAMA
1.0E-1	**1.000**	0.063	0.472	0.003	0.996	0.918	**1.000**	**1.000**
1.0E-2	0.063	0.063	0.298	0.000	0.847	0.635	0.999	**1.000**
1.0E-3	0.000	0.063	0.199	0.000	0.386	0.189	0.972	**0.999**
1.0E-4	0.000	0.063	0.134	0.000	0.136	0.050	**0.848**	0.830
1.0E-5	0.000	0.063	0.102	0.000	0.054	0.014	0.662	**0.669**
1.0E-6	0.000	0.063	0.079	0.000	0.024	0.005	0.467	**0.509**
1.0E-7	0.000	0.063	0.060	0.000	0.014	0.001	0.288	**0.355**
1.0E-8	0.000	0.063	0.049	0.000	0.009	0.001	0.156	**0.220**

creasing of accuracy level, the optimization task will become more challenging as the positions of optima should be more precisely identified. Consequently, the algorithm will become more difficult to locate the optima.

5. Conclusions

This work implements and reports a DE algorithm with niching competition, supporting archive and adaptive local search strategies for multimodal optimization. The niching competition strategy is proposed to competitively search the solution space with niches. The supporting archive strategy is designed with a dual purpose of helping maintain the potential optima recovered during evolution as well as facilitate the evolution of population. While, the adaptive local search strategy is developed for efficiently and properly improving the niching seeds. The experimental results reveal that our proposed method is able to consistently locate the optima in the solution space with high accuracy and outperform related methods. The results also confirm the significance of the proposed three strategies in helping properly search the multimodal space.

To extend the work further, several directions can be considered. Firstly, it is desirable to incorporate the devised niching competition strategy into other meta-heuristic methods, e.g., PSO, for MMOPs. Second, it would be interesting to employ other niching schemes to obtain the niches for our proposed method. In this regard, if the niches delivered by the schemes have various sizes, then the sizes should also be taken into account to design the potential evaluation function. Finally, it would be also interesting to adaptively employ multiple local searches to improve the solutions during evolution, thus improving the performance of the proposed method further.

References

[1] D. Z. Tan, W. N. Chen, J. Zhang, W. J. Yu, Fast pedestrian detection using multimodal estimation of distribution algorithms, in: Genetic and Evolutionary Computation Conference, Association for Computing Machinery, Inc, 2017, pp. 1248–1255. doi:10.1145/3071178.3071237.

[2] E. Pérez, M. Posada, A. Lorenzana, Taking advantage of solving the resource constrained multi-project scheduling problems using multi-modal genetic algorithms, Soft Computing 20 (5) (2016) 1879–1896. doi:10.1007/s00500-015-1610-z.

[3] C. H. Yoo, D. K. Lim, H. K. Jung, A novel multimodal optimization algorithm for the design of electromagnetic machines, IEEE Transactions on Magnetics 52 (3) (2016) 1–4. doi:10.1109/TMAG.2015.2478060.

[4] W. Gong, Y. Wang, Z. Cai, L. Wang, Finding multiple roots of nonlinear equation systems via a repulsion-based adaptive differential evolution,

IEEE Transactions on Systems, Man, and Cybernetics: Systems 50 (4) (2020) 1499–1513. doi:10.1109/TSMC.2018.2828018.

[5] Y. J. Gong, J. Zhang, Y. Zhou, Learning multimodal parameters: A bare-bones niching differential evolution approach, IEEE Transactions on Neural Networks and Learning Systems 29 (7) (2018) 2944–2959. doi:10.1109/TNNLS.2017.2708712.

[6] X. Li, M. G. Epitropakis, K. Deb, A. Engelbrecht, Seeking multiple solutions: An updated survey on niching methods and their applications, IEEE Transactions on Evolutionary Computation 21 (4) (2017) 518–538. doi:10.1109/TEVC.2016.2638437.

[7] Y. H. Zhang, Y. J. Gong, Y. Gao, H. Wang, J. Zhang, Parameter-free voronoi neighborhood for evolutionary multimodal optimization, IEEE Transactions on Evolutionary Computation 24 (2) (2020) 335–349. doi:10.1109/TEVC.2019.2921830.

[8] S. W. Mahfound, Crowding and preselection revisited, Parallel Problem Solving from Nature 2 (1992) 27–36.

[9] A. Della Cioppa, C. D. Stefano, A. Marcelli, Where are the niches? dynamic fitness sharing, IEEE Transactions on Evolutionary Computation 11 (4) (2007) 453–465. doi:10.1109/TEVC.2006.882433.

[10] C. Stoean, M. Preuss, R. Stoean, D. Dumitrescu, Multimodal optimization by means of a topological species conservation algorithm, IEEE Transactions on Evolutionary Computation 14 (6) (2010) 842–864. doi:10.1109/TEVC.2010.2041668.

[11] W. Gao, G. G. Yen, S. Liu, A cluster-based differential evolution with self-adaptive strategy for multimodal optimization, IEEE Transactions on Cybernetics 44 (8) (2014) 1314–1327. doi:10.1109/TCYB.2013.2282491.

[12] X. Li, Niching without niching parameters: Particle swarm optimization using a ring topology, IEEE Transactions on Evolutionary Computation 14 (1) (2010) 150–169. doi:10.1109/TEVC.2009.2026270.

[13] B. Y. Qu, P. N. Suganthan, J. J. Liang, Differential evolution with neighborhood mutation for multimodal optimization, IEEE Transactions on Evolutionary Computation 16 (5) (2012) 601–614. doi:10.1109/TEVC.2011.2161873.

[14] Y. H. Zhang, Y. J. Gong, W. N. Chen, Z. H. Zhan, J. Zhang, A generic archive technique for enhancing the niching performance of evolutionary computation, in: IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - SIS 2014: 2014 IEEE Symposium on Swarm Intelligence, 2015, pp. 1–8. doi:10.1109/SIS.2014.7011784.

[15] S. Kalra, S. Rahnamayan, K. Deb, Enhancing clearing-based niching method using delaunay triangulation, in: IEEE Congress on Evolutionary Computation, 2017, pp. 2328–2337. doi:10.1109/CEC.2017.7969587.

[16] S. Kundu, S. Biswas, S. Das, P. N. Suganthan, Crowding-based local differential evolution with speciation-based memory archive for dynamic multimodal optimization, in: Genetic and Evolutionary Computation Conference, 2013, pp. 33–40. doi:10.1145/2463372.2463392.

[17] B. Lacroix, D. Molina, F. Herrera, Region-based memetic algorithm with archive for multimodal optimisation, Information Sciences 367 (2016) 719–746. doi:10.1016/j.ins.2016.05.049.

[18] Y. Cao, H. Zhang, W. Li, M. Zhou, Y. Zhang, W. A. Chaovalitwongse, Comprehensive learning particle swarm optimization algorithm with local search for multimodal functions, IEEE Transactions on Evolutionary Computation 23 (4) (2019) 318–731. doi:10.1109/TEVC.2018.2885075.

[19] T. Huang, Y. J. Gong, S. Kwong, H. Wang, J. Zhang, A niching memetic algorithm for multi-solution traveling salesman problem, IEEE Transac-

tions on Evolutionary Computation 24 (3) (2020) 508–522. doi:10.1109/TEVC.2019.2936440.

[20] Z. Ren, A. Zhang, C. Wen, Z. Feng, A scatter learning particle swarm optimization algorithm for multimodal problems, IEEE Transactions on Cybernetics 44 (7) (2014) 1127–1140. doi:10.1109/TCYB.2013.2279802.

[21] Q. Yang, W. N. Chen, Y. Li, C. L. Chen, X. M. Xu, J. Zhang, Multimodal estimation of distribution algorithms, IEEE Transactions on Cybernetics 47 (3) (2017) 636–650. doi:10.1109/TCYB.2016.2523000.

[22] Z. J. Wang, Z. H. Zhan, Y. Lin, W. J. Yu, H. Wang, S. Kwong, J. Zhang, Automatic niching differential evolution with contour prediction approach for multimodal optimization problems, IEEE Transactions on Evolutionary Computation 24 (1) (2020) 114–128. doi:10.1109/TEVC.2019.2910721.

[23] G. Harik, Finding multimodal solutions using restricted tournament selection, in: International Conference on Genetic Algorithms, 1995.

[24] G. Singh, K. Deb, Comparison of multi-modal optimization algorithms based on evolutionary algorithms, in: Conference on Genetic & Evolutionary Computation, 2006, pp. 1305–1312.

[25] A. Linhares, Synthesizing a predatory search strategy for VLSI layouts, IEEE Transactions on Evolutionary Computation 3 (2) (1999) 147–152. doi:10.1109/4235.771168.

[26] A. Linhares, State-space search strategies gleaned from animal behavior: A traveling salesman experiment, Biological Cybernetics 78 (3) (1998) 167–173. doi:10.1007/s004220050423.

[27] Z. J. Wang, Z. H. Zhan, Y. Lin, W. J. Yu, H. Q. Yuan, T. L. Gu, S. Kwong, J. Zhang, Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems, IEEE Transactions on Evolutionary Computation 22 (6) (2018) 894–908. doi:10.1109/TEVC.2017.2769108.

[28] M. G. Epitropakis, X. Li, E. K. Burke, A dynamic archive niching differential evolution algorithm for multimodal optimization, in: IEEE Congress on Evolutionary Computation, 2013, pp. 79–86. doi:10.1109/CEC.2013.6557556.

[29] E. L. Yu, P. N. Suganthan, Evolutionary programming with ensemble of explicit memories for dynamic optimization, in: IEEE Congress on Evolutionary Computation, 2009, pp. 431–438. doi:10.1109/CEC.2009.4982978.

[30] J. E. Vitela, O. Castaños, A real-coded niching memetic algorithm for continuous multimodal function optimization, in: IEEE Congress on Evolutionary Computation, 2008, pp. 2170–2177. doi:10.1109/CEC.2008.4631087.

[31] J. C. Ni, L. Li, F. Qiao, Q. D. Wu, A novel memetic algorithm based on the comprehensive learning pso, in: IEEE Congress on Evolutionary Computation, 2012, pp. 1–8. doi:10.1109/CEC.2012.6256632.

[32] B. Y. Qu, J. J. Liang, P. N. Suganthan, Niching particle swarm optimization with local search for multi-modal optimization, Information Sciences 197 (2012) 131–143. doi:10.1016/j.ins.2012.02.011.

[33] H. Wang, I. Moon, S. Yang, D. Wang, A memetic particle swarm optimization algorithm for multimodal optimization problems, Information Sciences 197 (2012) 38–52. doi:10.1016/j.ins.2012.02.016.

[34] Y. G. Petalas, K. E. Parsopoulos, M. N. Vrahatis, Memetic particle swarm optimization, Annals of Operations Research 156 (1) (2007) 99–127. doi:10.1007/s10479-007-0224-y.

[35] J. Kennedy, Particle swarm: Social adaptation of knowledge, in: IEEE Conference on Evolutionary Computation, 1997, pp. 303–308. doi:10.1109/icec.1997.592326.

[36] F. J. Solis, R. J. Wets, Minimization by random search techniques, Mathematics of Operations Research 6 (1) (1981) 19–30. doi:10.1287/moor.6.1.19.

[37] A. Sharifi, J. K. Kordestani, M. Mahdaviani, M. R. Meybodi, A novel hybrid adaptive collaborative approach based on particle swarm optimization and local search for dynamic optimization problems, Applied Soft Computing 32 (2015) 432–448. doi:10.1016/j.asoc.2015.04.001.

[38] N. Hansen, S. D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es), Evolutionary Computation 1 (11) (2003) 1–8. doi:10.1162/106365603321828970.

[39] A. Caponio, F. Neri, V. Tirronen, Super-fit control adaptation in memetic differential evolution frameworks, Soft Computing 13 (8) (2009) 811–831. doi:10.1007/s00500-008-0357-1.

[40] V. Tirronen, F. Neri, T. Kärkkäinen, K. Majava, T. Rossi, An enhanced memetic differential evolution in filter design for defect detection in paper production, Evolutionary Computation 16 (4) (2008) 529–555. doi:10.1162/evco.2008.16.4.529.

[41] A. Caponio, G. L. Cascella, F. Neri, N. Salvatore, M. Sumner, A fast adaptive memetic algorithm for online and offline control design of pmsm drives, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 37 (1) (2007) 28–41. doi:10.1109/TSMCB.2006.883271.

[42] R. D. Al-Dabbagh, F. Neri, N. Idris, M. S. Baba, Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy, Swarm and Evolutionary Computation 43 (2018) 284–311. doi:10.1016/j.swevo.2018.03.008.

[43] F. Neri, C. Cotta, Memetic algorithms and memetic computing optimiza-

tion: A literature review, Swarm and Evolutionary Computation 2 (2012) 1–14. doi:10.1016/j.swevo.2011.11.003.

[44] X. Chen, Y. S. Ong, M. H. Lim, K. C. Tan, A multi-facet survey on memetic computation, IEEE Transactions on Evolutionary Computation 15 (5) (2011) 591–607. doi:10.1109/TEVC.2011.2132725.

[45] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, in: IEEE Congress on Evolutionary Computation, 2013, pp. 71–78. doi:10.1109/CEC.2013.6557555.

[46] S. Biswas, S. Kundu, S. Das, Inducing niching behavior in differential evolution through local information sharing, IEEE Transactions on Evolutionary Computation 19 (2) (2015) 246–263. doi:10.1109/TEVC.2014.2313659.

[47] M. Wang, B. Li, G. Zhang, X. Yao, Population evolvability: Dynamic fitness landscape analysis for population-based metaheuristic algorithms, IEEE Transactions on Evolutionary Computation 22 (4) (2018) 550–563. doi:10.1109/TEVC.2017.2744324.

[48] X. Li, A. Engelbrecht, M. Epitropakis, Benchmark functions for cec'2013 special session and competition on niching methods for multimodal function optimization, Tech. rep., Evol. Comput. Mach. Learn. Group., RMIT Univ., Melbourne, VIC, Australia (2013).

[49] C. Wang, R. Luo, K. Wu, B. Han, A new filled function method for an unconstrained nonlinear equation, Journal of Computational and Applied Mathematics 235 (6) (2011) 1689–1699. doi:10.1016/j.cam.2010.09.010.

[50] R. Thomsen, Multimodal optimization using crowding-based differential evolution, in: IEEE Congress on Evolutionary Computation, Vol. 2, 2004, pp. 1382–1389. doi:10.1109/cec.2004.1331058.