Towards Fast Niching Evolutionary Algorithms: A Locality Sensitive Hashing-Based Approach

Yu-Hui Zhang, *Student Member, IEEE*, Yue-Jiao Gong, *Member, IEEE*, Hua-Xiang Zhang, *Member, IEEE*, Tian-Long Gu, *Member, IEEE*, and Jun Zhang, *Senior Member, IEEE*

***Abstract*—Niching techniques have recently been incorporated into evolutionary algorithms for multi-solution optimization in multimodal landscape. However, existing niching techniques inevitably increase the time complexity of basic evolutionary algorithms due to the computation of the distance matrix of individuals. In this paper, we propose a fast niching technique. The technique avoids pairwise distance calculations by introducing the Locality Sensitive Hashing, an efficient algorithm for approximately retrieving nearest neighbors. Individuals are projected to a number of buckets by hash functions. The similar individuals possess a higher probability of being hashed into the same bucket than the dissimilar ones. Then, interactions between individuals are limited to the candidates that fall in the same bucket to achieve local evolution. It is proved that the complexity of the proposed fast niching is linear to the population size. In addition, this mechanism induces stable niching behavior and it inherently keeps a balance between the exploration and exploitation of multiple optima. The theoretical analysis conducted in the paper suggests that the proposed technique is able to provide bounds for the exploration and exploitation probabilities. Experimental results show that the fast niching versions of the multimodal algorithms can exhibit similar or even better performance than their original ones. More importantly, the execution time of the algorithms is significantly reduced.**

***Index Terms*—Evolutionary algorithm, locality sensitive hashing, multimodal optimization, niching technique, random projections.**

I. Introduction

MUTIMODAL optimization problems refer to problems that have multiple satisfactory solutions. These problems are frequently encountered in a variety of real-world applications. Some typical examples include: electromagnetic devices design [1], job shop scheduling [2], document retrieval [3], verification of grounding systems [4], and multiple ellipses detection [5]. When solving such problems, it is beneficial to have multiple good solutions located. First, it may happen that a solution cannot be achieved due to physical constraints or the occurrence of an emergency. In this case, if multiple solutions are known, we can quickly switch to other solutions. Second, multiple solutions can provide useful domain knowledge of the problem being solved. In this way, *Multimodal Optimization* has become a new research field, which requires a search algorithm to simultaneously locate multiple optima of a given problem. This field has received increasing attention recently because of its scientific and technological significance.

Evolutionary algorithms (EAs) [6], such as genetic algorithms (GAs) [7], differential evolution (DE) [8], and particle swarm optimization (PSO) [9], are natural candidates for solving multimodal optimization problems. This owes to the fact that they are population-based search algorithms. During the optimization process, an EA maintains a population of individuals (candidate solutions). The individuals evolve iteratively via genetic operators or interactive learning behaviors. If multiple distinct solutions can be preserved from iteration to iteration, then at the completion of the algorithm, there will be multiple good solutions in the final generation. From this viewpoint, EAs have distinctive advantages over traditional optimization approaches that require multiple runs for multiple optima. However, it is important to note that EAs are originally designed to locate a single global optimum. Without special treatment, the population of an EA will converge to a single solution.

To tackle multimodal problems with EAs, special techniques that can support multiple convergence are developed. The techniques are generally referred to as *niching* [10], [11]. They are incorporated into EAs to prevent convergence to a single solution, and to encourage individuals to find and preserve multiple *niches*. Some early niching techniques include fitness sharing [12], crowding [13], restricted tournament selection (RTS) [14], and speciation [15]. Although these techniques have been demonstrated to be effective in solving several multimodal functions, their performance is very sensitive to the setting of niching parameters (e.g., the sharing radius of the fitness sharing method and the speciation radius of the speciation method). In many cases, these parameters are very difficult to set without *a prior* knowledge. Besides the difficulty of setting niching parameters, other identified issues of multimodal optimization include the difficulty of detecting and maintaining optima in high dimensional space and the difficulty of balancing between exploitation and exploration. Current research on multimodal optimization attempts to resolve these problems by using distance-based neighborhoods to form different species in the search space [16]–[18]. Most recent studies have demonstrated that neighborhood information is crucial to the enhancement of niching performance [19]–[21]. However, there are still some drawbacks of applying the method. First, it incurs extra computational time overhead. To find the neighbors of each individual, we need to calculate the

This work was supported by the National Natural Science Foundation of China under Grant 6120002, 61502542, and U1201258.

Y.-H. Zhang, Y.-J. Gong, and J. Zhang are with School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. (email: gongyuejiao@gmail.com, junzhang@ieee.org)

Y.-H. Zhang is also a Ph.D. candidate with Sun Yat-sen University, Guangzhou, China.

H.-X. Zhang is with Shandong Normal University, Jinan, China.

T.-L. Gu is with Guilin University of Electronic Technology, Guilin, China.

pairwise distances of individuals. This significantly increases the time complexity of basic EAs. Second, it impairs the exploration ability of EAs. Using neighborhood information, offspring generated by an EA tend to gravitate toward inner regions enclosed by the initial population. Consequently, the number of optima located will rely likely on the initial distribution of individuals.

To alleviate the above problems, in this paper, we propose a fast niching technique. The goal is to reduce the time complexity of evolutionary multimodal algorithms while at the same time improving their performance. The fast niching technique is based on the theory of locality sensitive hashing (LSH) [22], [23]. The key idea is to avoid the running time bottleneck by using approximation methods to retrieve nearest neighbors. Meanwhile, the proposed technique increases the probability of interactions between distant individuals. This feature is helpful to the production of exploratory offspring, which restores the exploration ability of EAs. In the paper, we show that the technique can give both an upper bound and a lower bound to the exploration and exploitation probabilities respectively. To summarize, the advantages of the proposed niching technique over previous methods include: (1) it preserves the time complexity of basic EAs; (2) it is able to find a balance between exploitation and exploration automatically; and (3) it can be incorporated into different kinds of EAs. Experimental results show that, by integrating our LSH-based niching technique, the multimodal EAs can achieve improved efficiency and effectiveness.

The remainder of this paper is organized as follows. Section II gives a review of the existing multimodal algorithms. Section III introduces the locality sensitive hashing (LSH) method. Then, the proposed fast niching technique and its related theory are detailed in Section IV. Experiments are conducted in Section V, with comprehensive analysis of the numerical results. Finally, Section VI concludes this paper and provides some promising future research directions.

II. Related Work

Recently, various kinds of niching techniques have been developed and incorporated into EAs to tackle multimodal optimization problems. The maintenance of population diversity is one of the major challenges for the design of niching techniques. In the following, we divide the niching techniques into three groups according to the way the population diversity is maintained. For each group, we briefly review a few representative niching approaches and give a short discussion on their time complexity.

A. Niching by Modifying Replacement Mechanism

The traditional replacement mechanisms of EAs (e.g., tournament selection [7], roulette-wheel selection [24]) relies solely on the fitness values of individuals. Using these selection operators, individuals with better fitness values will have more chances to survive for the next iteration. They are efficient when searching for a global optimum. However, when dealing with multimodal problems, multiple optima often need to be detected simultaneously. If the greedy selection operators are used, it is probable that all the selected individuals are in the same region of attraction. Therefore, some methods attempt to preserve the population diversity by modifying the replacement mechanism of EAs. This group of methods includes crowding, restricted tournament selection, fitness sharing, and clearing. Some recently proposed bi-objective optimization techniques [25], [26] also fall in this category.

1) Crowding: Crowding was proposed by De Jong [13]. The idea is that similar individuals in the population must compete with each other for limited resources. An offspring is compared with CF (a parameter called crowding factor) individuals randomly sampled from the population. The nearest individual to the offspring will be replaced if its fitness is worse than that of the offspring. This approach is conducive to the maintenance of the diversity of the whole population. However, it suffers from the problem of replacement errors. To alleviate the problem, Mahfoud proposed a deterministic crowding method [10] which eliminates the parameter CF. An offspring is compared with its nearest neighbor in the population. In a subsequent study, Thomsen [16] integrated the crowding method with DE and proposed a crowding DE (CDE) algorithm.

2) Fitness sharing: Fitness sharing is a well-known niching method proposed by Holland [27]. It is later extended by Goldberg and Richardson [12]. The idea is that an individual needs to share its fitness value with other individuals close to it. More specifically, the shared fitness value of individual $\boldsymbol{x}_i$ is obtained by dividing the original fitness value by the number of individuals that are sufficiently close to $\boldsymbol{x}_i$. In a subsequent study, Thomsen proposed a sharing DE (ShDE) algorithm [28] based on the idea of fitness sharing.

3) Clearing: Clearing was proposed by Petrowski [29]. Different from crowding and fitness sharing, the clearing method identifies the best individual in a region of attraction and removes all other individuals in the vicinity of the best individual. In this way, similar individuals in the population are eliminated and resources are allocated to diverse offspring.

B. Niching by Modifying Reproduction Strategy

The robustness of population-based search methods is attributed to the interactions between individuals that share information about the problem landscape. The distribution of offspring generations of an EA depends largely on the way of interaction. For example, the mutation strategies of DE define the way of producing donor vectors. Similarly, in PSO, the moving trajectory of particles is influenced by their communication topology. However, traditional reproduction strategies are designed for finding a single optimal solution. They are not suitable for the task of locating multiple optima. For instance, the global communication topology of PSO will guide the particles fly towards a single solution located by the global best particle $\boldsymbol{gBest}$. Hence, the second group of niching techniques attempts to induce niching behavior by modifying the reproduction strategies of EAs.

Li [30] proposed a fitness Euclidean-distance ratio-based PSO (FER-PSO). The particles are guided towards their "fittest-and-closest" neighbors. Li [28] also demonstrated that PSO

with a ring topology (rpso) is able to simultaneously locate multiple optima without using any niching parameters. In [31], Otani *et al.* proposed a DE variant called DE/isolated/1. The algorithm finds an isolated individual in the current population and produces new individuals around the isolated individual. By doing so, computational resources can be allocated to the detection of multiple optima in the search space. In [32], Epitropakis *et al.* proposed two new mutation strategies to improve the niching ability of DE. The two mutation strategies (called DE/nrand/1 and DE/nrand/2 respectively) use the nearest neighbor of the target vector as the base vector. To reduce the computational cost of the DE/nrand family, Epitropakis *et al.* [33] suggest to find the nearest neighbor of each individual in an index-based neighborhood. The modified mutation strategies are called DE/inrand/1 and DE/inrand/2 respectively. Qu *et al.* [19] proposed a neighborhood mutation strategy to eliminate the inefficiency of traditional mutation strategies for multimodal optimization. In the neighborhood mutation strategy, a donor vector $\boldsymbol{v}_i$ is generated as:

$$\boldsymbol{v}_i = \boldsymbol{x}_{r1} + F \times (\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3}) \tag{1}$$

where the vectors $\boldsymbol{x}_{r1}$, $\boldsymbol{x}_{r2}$, and $\boldsymbol{x}_{r3}$ are chosen from the m nearest neighbors of the target vector $\boldsymbol{x}_i$. The mutation strategy is integrated with three DE variants (CDE, SDE, and ShDE) to solve multimodal problems. Qu *et al.* [20] also proposed a locally informed particle swarm optimizer (LIPS). Instead of using $\boldsymbol{gBest}$ to guide the search of particles, LIPS uses several local bests. The velocity of a particle is updated by using the information provided by its neighbors. Specifically, the velocity of the i-th particle is updated as follows:

$$\boldsymbol{v}_i = \omega \times (\boldsymbol{v}_i + \varphi(\boldsymbol{P}_i - \boldsymbol{x}_i)) \tag{2}$$

where the exemplar $\boldsymbol{P}_i$ is defined as:

$$\boldsymbol{P}_i = \frac{\sum_j^{nsize}(\varphi_j \cdot \boldsymbol{nBest}_j)}{\varphi} \tag{3}$$

φ_j is a random number in the range $[0, 4/nsize]$ and φ is the summation of φ_j. $\boldsymbol{nBest}_j$ is the j-th nearest neighbor to $\boldsymbol{pBest}_i$. $nsize$ is the neighborhood size. The pseudo codes of LIPS and the neighborhood based CDE (NCDE) can be found in [20] and [19] respectively. These two algorithms are employed as baseline algorithms of our work. In [34], Biswas *et al.* developed a parent-centric normalized neighborhood mutation operator to facilitate the tracking and maintenance of optima. It is incorporated into a proximity-based crowding DE and the algorithm is called PNPCDE.

C. Multi-population, Speciation and Clustering

The above two groups of methods are implicit niching methods. Subpopulations are formed automatically after several successive iterations. Different from the above methods, the third group of methods explicitly divides the population into multiple subpopulations. Each subpopulation is responsible for finding one optimum. Evolutionary operators (e.g., crossover and mutation) are performed within each subpopulation. In addition, new operators are devised to avoid multiple subpopulations converge to one optimum. By using these new operators, multiple subpopulations can work cooperatively to locate different optima. Several terms (e.g., species, nation, and cluster) have been used in the literature as alternatives to the term *subpopulation*. They can be used interchangeably in the context of multimodal optimization.

The idea of speciation [15], [35] is to divide the population into a number of species according to the distance information. Each species is formed around a dominant individual called species seed. A species seed is first identified. Then, individuals whose distance to the species seed is less than a threshold value r_s are assigned to the species. In later studies, Li *et al.* embedded the concept of speciation into PSO and DE, and proposed SPSO [18] and SDE [17] respectively. Brits *et al.* [36] proposed a PSO variant called NichePSO. Multiple sub-swarms are spawned from the main swarm to locate multiple optima. Two sub-swarms will be merged if they intersect. Ursem proposed a multinational EA [37]. It uses a fitness-topology function called hill-valley to check whether two individuals are in the same region of attraction and divides the population into multiple nations. Rigling and Moore [38] use DE to evolve multiple subpopulations. An individual will be punished if it is too close to another individual that belongs to a different subpopulation. Zaharie [39] proposed a multi-resolution multi-population DE (MMDE). MMDE divides the population into equally sized subpopulations. A migration operator is devised to avoid premature convergence.

Similar to the speciation method, clustering-based niching techniques [40]–[45] use clustering algorithms (e.g., k-means [44], DBSCAN [42]) to capture the clustering tendency of individuals. The reproduction and selection operators are then restricted to the detected clusters. Gao *et al.* [40] proposed a clustering-based multi-population strategy for DE to handle multimodal problems. In [44], Passaro and Starita proposed a k-means PSO (kPSO), in which the communication topology of particles is constructed based the outcome of the standard k-means algorithm. A fuzzy clustering technique is used in [45] to improve the performance of GA on multimodal optimization. In [41], Qing *et al.* proposed a crowding clustering GA (CCGA) that incorporates the crowding and clustering techniques.

D. Time Complexity of Existing Niching Techniques

As mentioned earlier, the Euclidean distance information plays a very important part in the niching behavior of EAs. Most of the niching techniques described above involve a large number of distance calculations, which significantly increase the time complexity of basic EAs. The time overhead incurred by the niching techniques will limit their use to applications with real-time processing requirements. The time complexities of the multimodal algorithms are summarized in Table I. It is worth mentioning that the time complexity of standard GA, PSO, and DE is $O(D \cdot N_p)$, where D is the problem dimension and N_p the population size. As can be observed from Table I, only few algorithms (e.g., rpso and DE/inrand/1) are able to maintain this time complexity. Although rpso and DE/inrand can run very fast, their performance on complex multimodal problems is relatively unsatisfactory according to the literature.

TABLE I
TIME COMPLEXITY OF MULTIMODAL ALGORITHMS

Category	Method	Time Complexity
Modification of selection mechanism	Crowding (CDE [16])	$O(D \cdot {N_p}^2)$
	Clearing [29]	$O(D \cdot {N_p}^2)$
	Fitness sharing [27]	$O(D \cdot {N_p}^2)$
	Bi-objective methods (MOBiDE [25])	$O(D \cdot {N_p}^2)$
	...	...
Modification of reproduction strategy	Ring topology PSO [28]	$O(D \cdot N_p)$
	DE/nrand/1 [32]	$O(D \cdot {N_p}^2)$
	DE/inrand/1 [33]	$O(D \cdot N_p)$
	DE/isolated/1 [31]	$O(D \cdot {N_p}^2)$
	Neighborhood mutation [19] (NCDE, NSDE, NShDE)	$O(D \cdot {N_p}^2 + N_p \log N_p)$
	FER-PSO [30]	$O(D \cdot {N_p}^2)$
	LIPS [20]	$O(D \cdot {N_p}^2)$
	PNPCDE [34]	$O(D \cdot {N_p}^2)$
	...	...
Multi-population, speciation, and clustering	Speciation (SCGA [15],SPSO [18],SDE [17])	$O(D \cdot {N_p}^2)$
	NichePSO [36]	$O(D \cdot {N_p}^2)$
	Clustering (Self-CCDE, Self-CSDE [40])	$O(D \cdot {N_p}^2)$
	*k*PSO [44]	$O(D \cdot {N_p}^2)$
	CCGA [41]	$O(D \cdot {N_p}^2)$
	...	...

To realize fast and competitive multimodal algorithms, in this paper, we propose a fast niching technique. The detailed description of the fast niching technique will be presented in Section IV. In the following section, we briefly describe a method called locality sensitive hashing, the theory of which is the foundation of the proposed work.

III. LOCALITY SENSITIVE HASHING

A. Basic Principle

Locality sensitive hashing (LSH) [22] is a method originally designed to tackle similarity search problems. A similarity search problem is stated as follows. Suppose we have a collection of objects (e.g., documents or images), each characterized by a feature vector in a D-dimensional space. Given query objects in the form of coordinates in the space, we are required to retrieve the most similar object to each of the queries. If the feature space is high-dimensional, the current space partition-based techniques are only slightly faster than the brute-force algorithm that compares the query object to all the objects in the collection. Using approximation methods is helpful to overcome the time bottleneck, and, in many cases, approximate solutions are as good as the exact ones [23]. LSH is a powerful method based on the above consideration. The key idea is to hash objects (feature vectors) to buckets using a family of hash functions so as to ensure that the probability of collision is much higher for similar objects than for those dissimilar. Then, we can find the nearest neighbor of a query object by searching the bucket to which the object is hashed.

To illustrate the basic idea of LSH, we give an example in a two-dimensional Euclidean space [46]. The hash function family H in the example is defined as the set of one-dimensional projected planes, i.e., random lines, in the space. Each hash function in the family corresponds to a line, which has been divided into segments of equal length r. Each segment stands for a "bucket" for storing the objects (data points). In Fig. 1, the gray lines are used to denote the hash functions. There are infinite hash functions in H. If we take a hash function (line) randomly from the family and project the data points onto the line, then two close points will very likely be hashed into the same bucket. Indeed, according to the LSH theory [22], there is a guarantee that the probability of collision is higher for points that are close to each other than for those far apart. This property can be used to reduce the time complexity of finding nearest neighbors. Suppose there are five data points (A, B, C, D, and E) in the space. The process of finding the nearest neighbor of A is illustrated as follows. We take three hash functions h_1, h_2, and h_3 randomly from the hash function family. Then we compute the hash values of the data points under the three hash functions. It can be seen that under h_1, B and E have the same hash value as A. Under h_2 and h_3, only B has the same hash value as A. Based on the hash outcomes, we are going to find the nearest neighbor of A from the set $\{$B, E$\}$. In this way, the calculation of the distances between A and the points in $\{$C, D$\}$ is avoided. Although the example is given in two dimensional space, the method can be generalized to higher dimensional space. One thing to be noted is that the method described above is an approximation method. Two points that are far apart from each other may sometimes be hashed into one bucket. Suppose we are going to find the nearest neighbor of E and the hash functions randomly sampled from the family are still h_1, h_2, and h_3. If we go through the process again, it will return that B is the closest neighbor to E.

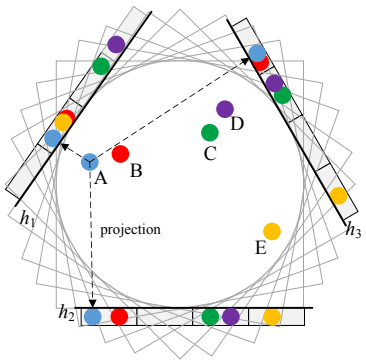

Fig. 1. Illustration of locality sensitive hashing (LSH) in two-dimensional Euclidean space.

B. Theory of LSH

The LSH method relies on the construction of a family of hash functions. An effective family of hash functions needs to meet the following three requirements [46]. (1) They must be more likely to produce the same hash codes for close pairs of points than for distant pairs. (2) The hash functions must be statistically independent. (3) They must be computationally efficient. The first requirement is associated with an important property of LSH families. Let S be the space of feature vectors with distance measure D, d_1 and d_2 are two distances

according to the distance measure, we have the following definition:

Definition 1: A family of hash functions H is said to be (d_1, d_2, p_1, p_2)-sensitive if for any $\boldsymbol{x}, \boldsymbol{y} \in S$:

$$\begin{cases} \Pr_{h\sim H}[h(\boldsymbol{x}) = h(\boldsymbol{y})] \geq p_1, & \text{if } D(\boldsymbol{x}, \boldsymbol{y}) \leq d_1 \\ \Pr_{h\sim H}[h(\boldsymbol{x}) = h(\boldsymbol{y})] \leq p_2, & \text{if } D(\boldsymbol{x}, \boldsymbol{y}) \geq d_2 \end{cases} \tag{4}$$

In the definition, the symbol $h \sim H$ has the meaning "randomly sample a hash function h from the family H". H is a valid LSH family when the inequalities $d_1 < d_2$ and $p_1 > p_2$ hold. Suppose there are two points to be processed. $\mathbf{E}$ denotes the event that the two points fall into the same bucket. The definition can be interpreted as follows. If the distance between the two points is less than d_1, then the probability of $\mathbf{E}$ will be at least p_1. On the other hand, if the distance is greater than d_2, then the probability of $\mathbf{E}$ will not exceed p_2.

The second requirement allows us to amplify the gap between the 'low' probability p_1 and the 'high' probability p_2 by concatenating several hash functions. Given an LSH family H, we can define a new hash family H' by using the *AND* operation.

Definition 2: For a fixed integer k, each member h of H' is in the form of $(h_1, h_2, \ldots, h_k)$, where h_i is a hash function chosen from H, independently and uniformly at random. Given two points $\boldsymbol{x}$ and $\boldsymbol{y}$, $h(\boldsymbol{x}) = h(\boldsymbol{y})$ if and only if $h_i(\boldsymbol{x}) = h_i(\boldsymbol{y})$ for every $i \in \{1, 2, \ldots, k\}$.

Since each member of H' is constructed by k independent members of H, we have the following theorem:

Theorem 1: If H is (d_1, d_2, p_1, p_2)-sensitive, then H' is (d_1, d_2, p_1^k, p_2^k)-sensitive.

The third requirement guarantees that the LSH method can achieve a considerable speedup with little comprise in solution quality. Due to its efficiency, LSH has found a number of applications in different fields, such as entity resolution, matching fingerprints, and matching newspaper articles [46].

IV. Fast Niching Technique

In this section, we exploit the potential of using the LSH for designing a fast niching technique that can induce stable niching behavior. After a detailed description of the fast niching technique, two fast multimodal algorithms are derived by integrating the technique with two recently proposed multimodal algorithms. The theoretical property of the fast niching technique is discussed subsequently. The computational complexity is analyzed at the end of this section.

A. Niching Using LSH

Traditional niching techniques involve a large number of distance calculations, which are very time consuming. The aim of these calculations is to find the nearest neighbors of each individual. This is because neighborhood information plays a very important role in the niching behavior. It is very likely that an individual will eventually converge to an optimum nearby if it interacts with its neighbors. The use of neighborhood information greatly enhances the exploitation ability of EAs. However, the excessive use of neighborhood information can also take away the exploration ability of EAs.

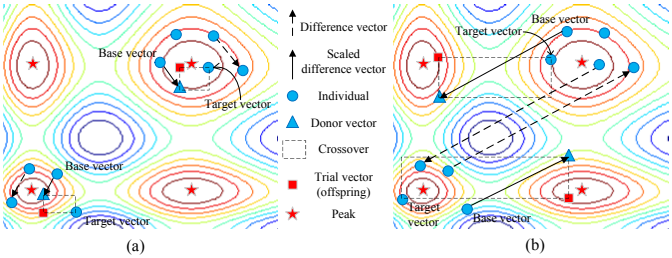

Fig. 2. Niching using LSH. (a) Interactions between close individuals. (b) Interactions between distant individuals.

As pointed out by Hui and Suganthan [47], one key challenge of evolutionary multimodal optimization is to balance between local solution exploitation and global exploration. To summarize, two issues are identified in the design of a niching technique. (1) The considerable time overhead caused by distance calculations. (2) The difficulty of balancing between exploration and exploitation.

Little work has been reported on the refinement of the complexity of multimodal algorithms. The major goal of our fast niching technique is to reduce the time complexity and this is where LSH comes in. A family of hash functions in Euclidean space can be applied to the design of a niching technique that requires little computational time. The key idea is to project individuals in the current population onto several randomly generated lines. An individual is only allowed to interact with members that are hashed to the same bucket of a line. According to the property of LSH families, individuals in a bucket are mostly very close to each other. Compared with existing niching methods, this approach eliminates the need of distance calculations and have similar effects on the niching behavior of EAs. Furthermore, it has another appealing feature that can address the second issue described above. It inherently avoids the overuse of neighborhood information. For example, in DE, while generating a trial vector $\boldsymbol{u}_i$ for the target vector $\boldsymbol{x}_i$. If all the vectors $\boldsymbol{x}_{r1}$, $\boldsymbol{x}_{r2}$, and $\boldsymbol{x}_{r3}$ are chosen from $\boldsymbol{x}_i$'s neighbors, the trial vector generated will still be in the vicinity of $\boldsymbol{x}_i$. To generate more diverse offspring, we need to sample the vectors from a wider range of area. This requirement is easily met because the approach is based on an approximation scheme. Distant individuals may be projected to one bucket due to the existence of approximation errors. This potentially restores the exploration ability EAs.

To illustrate the point, an example is shown in Fig. 2. There are four regions of attraction. Only two of them have been detected by the individuals. The detected regions can be well exploited if neighborhood information is incorporated into the mutation operator of DE (illustrated in Fig. 2 (a)). To discover the other two regions of attraction, offspring that are far away from the vectors are desired (illustrated in Fig. 2 (b)). The LSH-based approach supports such kind of explorations by allowing the target vector to interact with distant individuals in the same bucket. While ensuring the exploitation, a sufficient amount of effort will be paid to explore unknown regions. In this way, we can have a balance between the exploitation and exploration abilities.

B. The Proposed Fast Niching Technique

The proposed fast niching technique belongs to the second group of niching methods reviewed in Section II-B. It is devoted to improving the way of generating offspring populations. The procedure of the fast niching technique is described as follows:

Step 1: Define an LSH family H. Randomly choose nh hash functions $\{h_1, h_2, \ldots, h_{nh}\}$ from H.

Step 2: For each $h_i \in \{h_1, h_2, \ldots, h_{nh}\}$, hash the individuals in the current population to a number of buckets according to h_i. For each bucket of h_i, record the indexes of individuals that are hashed to it. Meanwhile, for each individual, record its hash value (i.e., bucket ID) under h_i.

Step 3: For each individual, define a group of members for interaction. The members are chosen from the buckets which the individual is hashed to.

In the optimization process, Step 1 and Step 2 are executed every nh iterations to amortize the cost of the hash function computation (i.e., we resample the hash functions every nh iterations). Step 3 may vary according to the baseline algorithm we are going to integrate with. The three steps will be detailed in the following of this subsection.

1) Generating the LSH Family: There are two things that need to be taken into consideration in the design of a hash family. (1) The hash function family should be "locality sensitive" so that close individuals will have a high probability to be hashed into the same bucket. (2) The hash functions should be computationally efficient so that the hash values of individuals can be computed very quickly. In Step 1, a recently proposed LSH family based on p-stable distribution is employed [48].

Definition 3: A distribution is said to be p-stable if it satisfies the following condition. There exists $p \geq 0$ such that for any n real numbers $X_1, X_2, \ldots, X_n$ and n independent identically distributed (i.i.d.) variables $V_1, V_2, \ldots, V_n$ with distribution $\mathcal{D}$, the random variable $\sum_{i=1}^{n} X_i V_i$ has the same distribution as the variable $(\sum_{i=1}^{n} |X_i|^p)^{1/p} V$, where V is a random variable with distribution $\mathcal{D}$.

Gaussian distribution is one of the well-known p-stable distributions and it is used in our implementation. The idea of the p-stable-based LSH scheme is to generate a D-dimensional random vector $\boldsymbol{V}$ whose entries are sampled independently from a p-stable distribution. Given a vector $\boldsymbol{x}$, we calculate the dot product $(\boldsymbol{x}^T \boldsymbol{V})$. Then, this dot product is used to assign a hash value for $\boldsymbol{x}$. Intuitively, the hash function family defined in such way is locality sensitive. Note that the dot product projects a vector onto a line. For two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, the distance between their projections $(\boldsymbol{x}_1 - \boldsymbol{x}_2)^T \boldsymbol{V}$ is distributed as $(\sum_{i=1}^{D} |\boldsymbol{x}_1 - \boldsymbol{x}_2|^p)^{1/p} V$ (i.e., $\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_p V$, according to the definition of stable distributions). Therefore, if $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are close (i.e., $\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_p$ is small), then they should collide with a high probability. On the other hand, if they are far from each other, they should collide with a small probability [48].

Each individual in the population is represented by a D-dimensional vector $\boldsymbol{x}_i$. After projecting the individuals onto a line, we record the coordinates ($minh$ and $maxh$) of the two points at both ends of the line. Then, the bucket size r is calculated in the following way:

$$r = \frac{maxh - minh}{nb} \tag{5}$$

where nb is a parameter used to control the number of buckets. Subsequently, we divide the hash line into a number of equal-width segments of size r and assign hash values to the individuals based on which segments they project to. Formally, a hash function is defined as follows:

$$h_{\boldsymbol{V},b}(\boldsymbol{x}_i) = \left\lfloor \frac{\boldsymbol{x}_i^T \boldsymbol{V} + b}{r} \right\rfloor \tag{6}$$

where $\boldsymbol{V}$ is a D-dimensional vector whose entries are randomly sampled from the standard Gaussian distribution and b is a real number randomly generated within the interval $[0, r]$. Fig. 3 illustrates the effect of the random number b. Intuitively, b is used to shift the projections of individuals. Suppose that the absolute difference between the projections of two individuals is t. If t is larger than r, then the two individuals will definitely be in two different buckets. On the other hand, if t is less than r, then the probability of the two individuals being in the same bucket would be $1 - \frac{t}{r}$. In (6), $\lfloor \cdot \rfloor$ denotes the floor function. The hash function maps D-dimensional vectors onto the set of integers. Furthermore, we amplify the hash family by using an *AND* operation that concatenates two basic hash functions. Specifically, a hash function in the constructed LSH family can be formulated as:

$$h_{\boldsymbol{V}_1,b_1,\boldsymbol{V}_2,b_2}(\boldsymbol{x}_i) = Z \times \left\lfloor \frac{\boldsymbol{x}_i^T \boldsymbol{V}_1 + b_1}{r_1} \right\rfloor + \left\lfloor \frac{\boldsymbol{x}_i^T \boldsymbol{V}_2 + b_2}{r_2} \right\rfloor \tag{7}$$

where Z is a sufficiently large integer, $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ are mutually independent random vectors. One advantage of the hash functions is that they can be computed very efficiently.

2) Hashing the Individuals: In Step 2, for each hash function h_i in the form of (7), we project the individuals in the current population to a number of buckets. The bucket ID of an individual is given by its hash value. An auxiliary data structure is then used to store the hash outcomes. The data structure is designed to fast retrieve the contents of each bucket. To ensure efficiency, the bucket IDs of the individuals are stored in a simple array, while the storage of member IDs for the buckets is implemented as a dictionary (a collection of (key, value) pairs). When looking for individuals that are in the same bucket as $\boldsymbol{x}_i$, we first look up $\boldsymbol{x}_i$'s hash value (bucket ID) in the array. Then, the bucket ID (key) is used to

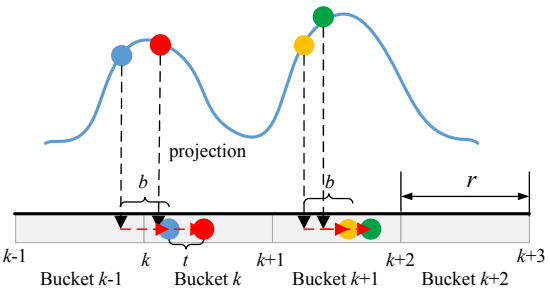

Fig. 3. Illustration of the effect of b.

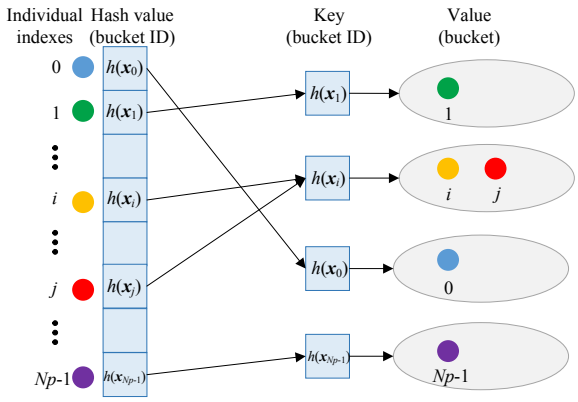

Fig. 4. Implementation of the fast niching technique.

locate the associated bucket storing the indexes of individuals. This process is plotted in Fig. 4. Since we have sampled nh functions from the hash function family, nh copies of the data structure are required to store all the hash outcomes. For each individual, we will obtain nh bucket IDs from the nh hash functions. The bucket IDs will remain unchanged until we resample the hash functions from the hash function family.

3) Within-Buckets Evolution: In Step 3, with the aid of the data structure, we can efficiently find the neighbors of each individual. One thing to be noted is that the way of utilizing the hashed individuals varies according to the different type of EA we are using. For PSO, the hashing relationships characterize the communication topology of particles. For DE, they are used for the purpose of selecting vectors $\boldsymbol{x}_{r1}$, $\boldsymbol{x}_{r2}$, and $\boldsymbol{x}_{r3}$.

In order to examine the effectiveness of the fast niching technique, we integrate it with two recently proposed multimodal algorithms, i.e., LIPS [20] and NCDE [19]. The resulting algorithms are named Fast-LIPS and Fast-NCDE, whose pseudo codes are given in **Algorithm 1** and **Algorithm 2**, respectively. The major difference between Fast-LIPS and LIPS is the formation of exemplar particles. Similarly, the difference between Fast-NCDE and NCDE is the identification of candidates for vectors $\boldsymbol{x}_{r1}$, $\boldsymbol{x}_{r2}$, and $\boldsymbol{x}_{r3}$. In NCDE, the vectors are chosen from the Euclidean neighbors of the target vector. In Fast-NCDE, they are chosen from the bucket to which the target vector is projected. The neighborhood size parameter m introduced by NCDE is eliminated by Fast-NCDE. Moreover, NCDE uses the crowding method that restricts the comparisons of offspring to their nearest neighbors in the current population. In Fast-NCDE, an offspring will compete with its nearest rival in the same bucket. Noting that in EAs evolution occurs through gradual modification of existing individuals, there is no need to change the bucket IDs too frequently. Therefore, to save computational effort, the hash functions are regenerated every nh iterations.

The fast niching technique introduces two parameters, i.e., the number of buckets (nb in (5)) and the number of hash functions (nh) used for niching. The effects of these parameters will be investigated in the experimental section. It is noteworthy that the fast niching technique can be integrated with different types of EAs. Moreover, it can be extended to any discrete multimodal optimization problems, as long as their distance metrics have an LSH family. In the following subsection, we give a brief discussion on the property of the LSH-based technique.

Algorithm 1 Fast-LIPS

$G = 0$; //G is the iteration number
Generate an initial swarm of particles $P = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{N_p}\}$ by uniformly and randomly sample N_p particles in the search space;
Randomly initialize the velocities $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{N_p}\}$ of the particles;
Evaluate the fitness of every particle in the initial swarm;
$FEs = N_p$; //FEs records the number of fitness evaluations
Record the personal best positions of the particles $PB = \{\boldsymbol{pBest}_1, \boldsymbol{pBest}_2, \ldots, \boldsymbol{pBest}_{N_p}\}$
while $FEs < MaxFEs$ **do**
  if G mod $nh == 0$ **then**
    Randomly generate nh hash functions $\{h_1, h_2, \ldots, h_{nh}\}$; // Step 1
    for $i = 1$ to nh **do**
      Project the members in PB to a number of buckets $BS_i = \{B_1, B_2, \ldots, B_{nb}\}$ using h_i; // Step 2
    end for
    for $i = 1$ to N_p **do**
      Randomly choose a bucket set BS_k, find the bucket B_w that contains $\boldsymbol{pBest}_i$; // Step 3
      Randomly choose $nsize$ members from B_w to form a group of $\boldsymbol{nBest}$s for the i-th particle;
    end for
  end if
  for $i = 1$ to N_p **do**
    Update the i-th particles velocity $\boldsymbol{v}_i$ using (2);
    Update the i-th particles position $\boldsymbol{x}_i$;
    Evaluate the fitness of $\boldsymbol{x}_i$;
    $FEs = FEs + 1$;
    if the fitness of $\boldsymbol{x}_i$ is better than that of $\boldsymbol{pBest}_i$ **then**
      $\boldsymbol{pBest}_i = \boldsymbol{x}_i$;
    end if
  end for
  $G = G + 1$;
end while

C. Theoretical Property of the Fast Niching Technique

To analyze the search behavior of the LSH-based niching algorithms, we first give the definitions of exploitation and exploration in the context of multimodal optimization.

Definition 4: Exploitation is defined as the interactions between individuals in the same region of attraction. By analogy, exploration is defined as the interactions between individuals from different regions of attraction.

The definitions are based on the consideration that an offspring is very likely to be in the same region as its parents if the parents are very close to each other. On the other hand, it is often the case that the offspring lies in a region different from both of its parents, if the parents are from two far

Algorithm 2 Fast-NCDE

$G = 0$; //G is the iteration number
Generate an initial population of individuals $P = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{N_p}\}$ by uniformly and randomly sample N_p individuals in the search space;
Evaluate the fitness of every individual in the initial population;
$FEs = N_p$;
while $FEs < MaxFEs$ **do**
  if G mod $nh == 0$ **then**
    Randomly generate nh hash functions $\{h_1, h_2, \ldots, h_{nh}\}$; // Step 1
    for $i = 1$ to nh **do**
      Project the individuals in the population to a number of buckets $BS_i = \{B_1, B_2, \ldots, B_{nb}\}$ using h_i; // Step 2
    end for
  end if
  for $i = 1$ to N_p **do**
    Randomly choose a bucket set BS_k, find the bucket B_w that contains $\boldsymbol{x}_i$; // Step 3
    Randomly choose three vectors, $\boldsymbol{x}_{r1}$, $\boldsymbol{x}_{r2}$, and $\boldsymbol{x}_{r3}$, from B_w;
    Generate an offspring (trial vector) $\boldsymbol{u}_i$ using DE's mutation and crossover operators;
    Evaluate the fitness of $\boldsymbol{u}_i$;
    $FEs = FEs + 1$;
    Randomly choose a hash function h_k, project $\boldsymbol{u}_i$ to a bucket using h_k, find the corresponding bucket B_w in the bucket set BS_k;
    Calculate the Euclidean distance of $\boldsymbol{u}_i$ to all other members in B_w;
    Find the nearest neighbor $\boldsymbol{x}_j$ in B_w to $\boldsymbol{u}_i$;
    if the fitness of $\boldsymbol{u}_i$ is better than $\boldsymbol{x}_j$ **then**
      $\boldsymbol{x}_j = \boldsymbol{u}_i$;
    end if
  end for
  $G = G + 1$;
end while

apart regions. The definitions are helpful to the description of the search behavior of the LSH-based approach. Given the definitions of exploration and exploitation, we have the following propositions.

Proposition 1: For any individual in the population, the LSH-based approach gives both a lower bound and an upper bound to its exploitation and exploration probabilities respectively.

Suppose there are m regions of attraction $\{R_1, R_2, \ldots, R_m\}$ that have been detected by the individuals. The distance between individuals in the same region of attraction is smaller than d_1 and the distances between individuals from different regions are larger than d_2. We use p_{xy} to denote the probability that two individuals $\boldsymbol{x}$ and $\boldsymbol{y}$ are hashed to the same bucket by a hash function drawn randomly from the hash family. In the proposed LSH-based technique, the partner of an individual is chosen randomly from the bucket in which the individual is

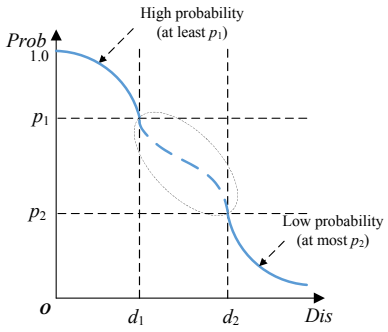

Fig. 5. Illustration of the (d_1, d_2, p_1, p_2)-sensitive family.

hashed. For an individual $\boldsymbol{x}$ in the ith region of attraction, its exploitation probability is calculated as:

$$p_{exploit} = \frac{\sum_{y \in R_i \setminus x} p_{xy}}{\sum_{y \in R_i \setminus x} p_{xy} + \sum_{j \in \{1,2,\ldots,m\} \setminus i} \sum_{y \in R_j} p_{xy}} \quad (8)$$

Similarly, the exploration probability is given by:

$$p_{explore} = \frac{\sum_{j \in \{1,2,\ldots,m\} \setminus i} \sum_{y \in R_j} p_{xy}}{\sum_{y \in R_i \setminus x} p_{xy} + \sum_{j \in \{1,2,\ldots,m\} \setminus i} \sum_{y \in R_j} p_{xy}} \quad (9)$$

Suppose there are four regions of attraction. For each region, there are 10 individuals located in it. $R(\boldsymbol{x})$ denotes the region of attraction to which $\boldsymbol{x}$ belongs. Suppose p_{xy} equals to 50% for $R(\boldsymbol{x}) = R(\boldsymbol{y})$ and 10% for $R(\boldsymbol{x}) \neq R(\boldsymbol{y})$. Then, according to (8) and (9), the exploitation and exploration probabilities will be 60% and 40% respectively. A slightly different example is that p_{xy} equals to 60% for $R(\boldsymbol{x}) = R(\boldsymbol{y})$ and 18% for $R(\boldsymbol{x}) \neq R(\boldsymbol{y})$. In this scenario, both exploitation and exploration probabilities will be 50%.

According to the definition of the LSH family, for distance less than d_1, the probability of collision is larger than p_1, and for distance larger than d_2, the probability of collision is less than p_2 (illustrated in Fig. 5). We can obtain the following bounds for $p_{exploit}$ and $p_{explore}$ respectively.

$$p_{exploit} \geq \frac{\sum_{y \in R_i \setminus x} p_1}{\sum_{y \in R_i \setminus x} p_1 + \sum_{j \in \{1,2,\ldots,m\} \setminus i} \sum_{y \in R_j} p_{xy}} \quad (10)$$

$$p_{explore} \leq \frac{\sum_{j \in \{1,2,\ldots,m\} \setminus i} \sum_{y \in R_j} p_2}{\sum_{y \in R_i \setminus x} p_{xy} + \sum_{j \in \{1,2,\ldots,m\} \setminus i} \sum_{y \in R_j} p_2} \quad (11)$$

Note that the definition of the LSH family gives no guarantee to p_{xy} if the distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ is within the range (d_1, d_2), we need the following proposition to make sure that the bounds will hold for varying d_1 and d_2.

Proposition 2: For any distances $d_1 < d_2$, the hash family based on the Gaussian distribution is (d_1, d_2, p_1, p_2)-sensitive for certain values p_1 and p_2.

Suppose the distance between two different individuals $\boldsymbol{x}$ and $\boldsymbol{y}$ is d. We can compute the probability of the event that $\boldsymbol{x}$ and $\boldsymbol{y}$ are hashed into the same bucket. From the construction of the LSH family, it can be seen that the difference between the projections of $\boldsymbol{x}$ and $\boldsymbol{y}$ is distributed as dV, where V is a random variable with the Gaussian distribution. Let $f_G(t)$ denote the probability density function of the standard

Gaussian distribution. The density function of dV is given by $\frac{1}{d}f_G(\frac{t}{d})$. Since b is a random number in the interval [0, r], the probability can be computed as:

$$p(d) = \Pr_{h\sim H}[h(\boldsymbol{x}) = h(\boldsymbol{y})] = 2\int_0^r \frac{1}{d} f_G(\frac{t}{d})(1-\frac{t}{r})\mathrm{d}t \tag{12}$$

For any bucket size r, the probability decreases monotonically with the distance d. In other words, for any two distances d_1 and d_2, as long as $d_1 < d_2$, the probability $p_1 = p(d_1)$ will be strictly larger than $p_2 = p(d_2)$.

From (10) and (11), it can be seen that the theoretical bounds are associated with the probabilities p_1 and p_2. Note that the two probabilities are affected by the parameter nb, we can use nb to adjust the ratios of exploration and exploitation. Specifically, when nb is small, the parameter r of the hash functions will become relatively large. Both p_1 and p_2 will increase in this case. Considering that the summation range $\sum_{j\in\{1,2,\ldots,m\}\setminus j}\sum_{y\in R_j}$ is larger than $\sum_{y\in R_i\setminus x}$, the upper bound of the exploration probability will increase more significantly than the lower bound of the exploitation probability. Hence, the ratio of exploration can be increased to some extent. Conversely, a large value of nb is able to increase the ratio of exploitation.

D. Complexity Analysis

The time complexity of evolutionary multimodal algorithms is governed by their niching components. Generally, it is estimated by the number of elementary operations performed at each iteration. Most of the existing niching techniques take $O(D \cdot {N_p}^2)$ time due to the requirement of pairwise distance calculations. This is not the case for our fast niching technique, whose time complexity is composed of three parts. In Step 1, generating nh hash functions takes $O(nh \cdot D)$ time. In Step 2, calculating hash values for individuals takes $O(nh \cdot D \cdot N_p)$ time, while building the data structures takes $O(nh \cdot N_p)$ time. In Step 3, the time complexity of finding the groups of members is $O(N_p)$. Hence, the total time complexity is $O(nh \cdot D \cdot N_p)$. Since the hash functions are resampled every nh iterations, the amortized cost of the fast niching technique for each iteration is $O(D \cdot N_p)$, which is the same as the basic operators of EAs. Therefore, incorporating the fast niching technique will not increase the time complexity of EAs.

V. Experimental Study

In this section, we carry out experiments to study the performance and running time of the multimodal algorithms developed using the fast niching technique. The effects of the parameters introduced by the fast niching technique are also investigated.

A. Experimental Setup

1) Test Functions: The benchmark function set for the CEC'2013 special section on multimodal optimization [49] is adopted to test the effectiveness of our multimodal algorithms. It is currently a state-of-the-art problem set and it contains 20 multimodal functions with different characteristics, which can be used to challenge different aspects of niching techniques. The test functions are tabulated in Table II. F1-F10 are frequently used functions in the multimodal optimization community. F11-F20 are complex multimodal functions constructed by several basic functions. All the test functions are maximization problems. The maximum number of fitness evaluations (MaxFEs) for each test function is given in the last column of Table II. Detailed descriptions about the test functions can be found in [49].

TABLE II
Test Functions

Function	Name	NPK	MaxFEs
F1	Five-Uneven-Peak Trap (1D)	2	5.00E+04
F2	Equal Maxima (1D)	5	5.00E+04
F3	Uneven Decreasing Maxima (1D)	1	5.00E+04
F4	Himmelblau (2D)	4	5.00E+04
F5	Six-hump Camel Back (2D)	2	5.00E+04
F6	Shubert (2D)	18	2.00E+05
F7	Vincent (2D)	36	2.00E+05
F8	Shubert (3D)	81	4.00E+05
F9	Vincent (3D)	216	4.00E+05
F10	Modified Rastrigin (2D)	12	2.00E+05
F11	Composition Function 1 (2D)	6	2.00E+05
F12	Composition Function 2 (2D)	8	2.00E+05
F13	Composition Function 3 (2D)	6	2.00E+05
F14	Composition Function 3 (3D)	6	4.00E+05
F15	Composition Function 4 (3D)	8	4.00E+05
F16	Composition Function 3 (5D)	6	4.00E+05
F17	Composition Function 4 (5D)	8	4.00E+05
F18	Composition Function 3 (10D)	6	4.00E+05
F19	Composition Function 4 (10D)	8	4.00E+05
F20	Composition Function 4 (20D)	8	4.00E+05

TABLE III
Algorithms Compared

No.	Algorithm	Description
1	CDE [16]	Crowding DE
2	SDE [17]	Speciation-based DE
3	ShDE [16]	Fitness sharing DE
4	DE/nrand/1 [32]	DE with a mutation strategy in which the base vector is the nearest neighbor of the target vector
5	DE/inrand/1 [33]	DE/nrand/1 with index-based nieghborhoods
6	NShDE [19]	Neighborhood mutation based ShDE
7	NSDE [19]	Neighborhood mutation based SDE
8	NCDE [19]	Neighborhood mutation based CDE
9	r2pso [28]	A local best PSO with the ring topology, each particle interacts with its immediate neighbor on its right;
10	r3pso [28]	A local best PSO with the ring topology, each particle interacts with its immediate neighbor on its left and right;
11	r2pso-lsh [28]	r2pso without overlapping neighborhoods
12	r3pso-lsh [28]	r3pso without overlapping neighborhoods
13	FER-PSO [30]	Fitness-Euclidean distance ratio PSO
14	LIPS [20]	Locally informed particle swarm optimizer
15	Fast-LIPS	Fast niching based LIPS
16	Fast-NCDE	Fast niching based NCDE

2) Performance Measure: Two popular performance measures, i.e., peak ratio (PR) and success rate (SR), are adopted to compare the performance of different multimodal algorithms [49].

(a) Peak ratio (PR) is the average percentage of global optima

found, which is calculated as:

$$PR = \frac{\sum_{i}^{NR} NPF_i}{NPK \times NR} \tag{13}$$

where NPF_i is the number of optima located in the i-th run. NR is the total number of runs. NPK is the number of global optima.

(b) Success rate (SR) is the percentage of successful runs in which all the global optima are located, which is given by:

$$SR = \frac{NSR}{NR} \tag{14}$$

where NSR denotes the number of successful runs.

3) Algorithms and Parameter Settings: To determine the advantages of the fast niching techniques, Fast-LIPS and Fast-NCDE are compared with a number of multimodal algorithms. In total, 16 algorithms are considered in the experiment, they are tabulated in Table III. The compared algorithms have covered all three groups of niching techniques. Besides the multimodal algorithms, the basic DE and PSO algorithms, which are originally designed to locate a single global optimum, are also used to check the effectiveness of the niching technique. The parameters of the algorithms are set according to the respective publications. The population size of the algorithms is fixed at 100. For Fast-LIPS, nb is set to 20. For Fast-NCDE, nb is set to 5. The number of hash functions used for both algorithms is set as $nh = 10$. All the algorithms are implemented in C++, and were executed on an Intel Xeon(R) X5675 computer with 3.07-GHz speed and 12 GB RAM. To obtain statistically reliable results, 50 independent runs are conducted for each test function, i.e., $NR = 50$. Moreover, a challenging accuracy level, namely 1E−04, is employed to compute the number of global optima found.

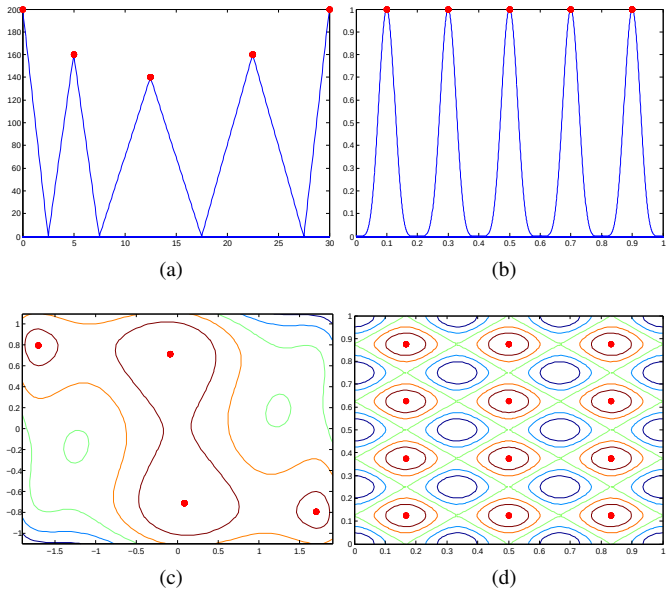

Fig. 6. Final generations of Fast-NCDE on low dimensional multimodal functions. (a) F1 (b) F2 (c) F5 (d) F10.

TABLE IV
RANKING OF THE MULTIMODAL ALGORITHMS BY THE FRIEDMAN'S TEST

Algorithm	Ranking
Fast-NCDE	2.65
Fast-LIPS	3.025
NCDE	4.25
DE/inrand/1	4.25
NShDE	4.75
LIPS	4.95
NSDE	6.025
r3pso-lsh	6.1

B. Experimental Results

Experimental results are summarized in Table S1. The best results with respect to PR are in bold. In addition, the Wilcoxon rank-sum test is adopted to detect whether there are significant differences between the results obtained by Fast-NCDE and those of other algorithms.

From Table S1, it can be observed that Fast-LIPS and Fast-NCDE can achieve higher PR values than other multimodal algorithms in a large portion of the test functions. Both of them achieve 100% PR and 100% SR for F1-F5, F10, which means that they have the ability to solve these problems consistently. F6-F9 are functions with many global optima. In particular F9 has a number of optima up to 216. It is very challenging for a population of 100 individuals to locate all the optima. Therefore, the SR values obtained by all the algorithms are not satisfactory. Nevertheless, Fast-LIPS and Fast-NCDE have made good use of the limited individuals and their PR values in these test functions are competitive. For complex composite multimodal functions with rugged landscapes, it is very difficult for all the algorithms to achieve a nonzero success rate. Hence, we concentrate on the PR values. According to the PR values of Fast-LIPS and Fast-NCDE, it can be inferred that they are able to converge to multiple optimal solutions. This provides evidence for the technique's capability of inducing niching behavior. In addition, the results of the non-parametric statistical tests have been included in the table. The superscript "#" denotes that the PR value of Fast-NCDE is significantly larger than that of the corresponding algorithm, while the subscript "b" indicates the opposite. The statistical results are summarized in the last row of the table. The comparisons show that Fast-NCDE significantly outperforms other algorithms in more than nine test functions. The final generations of Fast-NCDE on the low dimensional test functions are visualized in Fig. 6. It can be seen that Fast-NCDE successfully found the multiple optima of the test functions.

In order to determine the significant differences of the compared methods, the Friedman's test was conducted. The Bonferroni-Dunn was employed for the post-hoc test. Table IV provides the ranking of eight compared methods. As shown in Table IV, Fast-NCDE has the first ranking, closely followed by Fast-LIPS. In addition, the Friedman's test was carried out for the PSO-based and DE-based multimodal algorithms respectively. Table V summarizes the results. It can be observed that both Fast-LIPS and Fast-NCDE rank high in their respective group. Based on the above comparisons, we can

TABLE V
RANKING OF PSO-BASED AND DE-BASED MULTIMODAL ALGORITHMS BY THE FRIEDMAN'S TEST

Algorithm	Ranking	Algorithm	Ranking
Fast-LIPS	1.55	Fast-NCDE	2.6
LIPS	2.625	DE/nrand/1	3.775
r3pso-lsh	3.825	DE/inrand/1	3.975
r3pso	4.075	NCDE	4.15
r2pso-lsh	4.8	NShDE	4.475
r2pso	5.025	CDE	4.675
FER-PSO	6.1	NSDE	5.725
		SDE	7.225
		ShDE	8.4

TABLE VI
RESULTS OF THE MULTIPLE-PROBLEM WILCOXON'S TEST FOR FAST-LIPS AND LIPS

	ε	R+	R−	p-value	α=0.05	α=0.1
Fast-LIPS vs LIPS	1.00E−01	180	10	0.00058	Yes	Yes
	1.00E−02	180	10	0.00058	Yes	Yes
	1.00E−03	180	10	0.00058	Yes	Yes
	1.00E−04	180	10	0.00058	Yes	Yes
	1.00E−05	180	10	0.00058	Yes	Yes

TABLE VII
RESULTS OF THE MULTIPLE-PROBLEM WILCOXON'S TEST FOR FAST-NCDE AND NCDE

	ε	R+	R−	p-value	α=0.05	α=0.1
Fast-NCDE vs NCDE	1.00E−01	133.5	76.5	0.276717	No	No
	1.00E−02	126.5	63.5	0.195576	No	No
	1.00E−03	165	45	0.023255	Yes	Yes
	1.00E−04	175.5	34.5	0.007744	Yes	Yes
	1.00E−05	177.5	32.5	0.00168	Yes	Yes

see that the fast niching-based multimodal algorithms are very competitive.

C. Comparison with LIPS and NCDE

To further examine the effectiveness of the proposed technique, the fast niching multimodal algorithms and their original versions are compared pairwise. Five levels of accuracy, i.e., {1E−01, 1E−02, 1E−03, 1E−04, 1E−05}, are used to test the performance of these algorithms. Tables S2-S3 summarize the results of Fast-LIPS versus LIPS and Fast-NCDE versus NCDE respectively. Better PR values are emphasized in bold. In addition, the Wilcoxon rank sum test is applied to confirm the existence of significant differences. From Table S2, we can observe that Fast-LIPS consistently outperforms LIPS regardless of the setting of the accuracy level. According to the statistical tests, Fast-LIPS performs significantly better than LIPS in more than 12 test functions. The differences are more evident when dealing with high dimensional multimodal functions (F15-F20). Likewise, the fast niching technique can improve the performance of NCDE in a number of multimodal problems, as revealed in Table S3. Although Fast-NCDE is outperformed by NCDE in F20, there are at least six test functions in which Fast-NCDE has significantly better results. In addition, the gap between NCDE and Fast-NCDE is enlarged when the accuracy level raises to 1E−03, 1E−04, and 1E−05.

Table VI and Table VII collect the statistical test results of the multiple-problem Wilcoxon's test based on the PR values. As can be seen, both Fast-LIPS and Fast-NCDE achieve higher R+ values than R− values in all five levels of accuracy. From Table VI, it can be seen that the p values in all five levels of accuracy are less than 0.05, which suggests that Fast-LIPS has more reliable performance than LIPS. Similarly, according to the p values reported in Table VII, significant differences between Fast-NCDE and NCDE can be observed when the accuracy level is set to 1E−03, 1E−04, and 1E−05.

The above comparisons show that the fast niching-based multimodal algorithms have an edge over their original versions. This is attributed to the property that automatically keeps a balance between exploration and exploitation, which gives a boost to the performance of LIPS and NCDE.

1) Convergence speed: We move on to study the convergence speed of Fast-LIPS and Fast-NCDE. For the convergence speed of a multimodal algorithm we use the number of FEs required to locate all the optima. It is calculated as follows:

$$AvgFEs = \frac{\sum_{i=1}^{NR} FE_i}{NR} \tag{15}$$

where FE_i is the number of fitness evaluations consumed in the i-th run. If the algorithm cannot locate all the optima, then FE_i is given by the predefined MaxFEs. As suggested in [49], the accuracy level is fixed at 1E−04. The results on F1-F5 are listed in Table S4. For low dimensional functions with a few global optima, it is likely that each region of attraction will have several individuals scattered within it. Hence, algorithms that focus exclusively on exploitation will have a head start. From Table S4, it can be seen that LIPS has the fastest convergence speed. This is because LIPS uses a small number of local bests to guide the search of each particle and it greatly enhances the exploitation ability. However, the mechanism will also pose a heavy burden on the LIPS's performance on high dimensional functions or functions with many optimal solutions. Fig. 7 plots the number of global optima found against the number of FEs for F6, F10, F14, and F18. Although the convergence speed of Fast-LIPS is at first slower than that of LIPS, Fast-LIPS manages to surpass LIPS, as it can detect more regions of attraction. On the other hand, NCDE shows more exploratory behavior than LIPS since it uses a relatively large neighborhood size [19]. It is interesting to note that the convergence speed of Fast-NCDE is faster than that of NCDE, which suggests that Fast-NCDE has a good grasp of balance between exploitation and exploration.

2) Running Time of Fast-LIPS and Fast-NCDE: We proceed to compare the running time of the multimodal algorithms. The algorithms are compared pairwise to see the speedup provided by the fast niching technique. For clarity, the results are depicted in Fig. 8. As shown in the figures, the running time of LIPS and NCDE is substantially reduced. Fast-LIPS and Fast-NCDE are at least two times faster than LIPS and NCDE. Moreover, it can be observed that the running time increases as the number of dimensions grows. Besides the problem dimension, the time complexity of the algorithms is associated with the population size. In order to investigate the effect of population size, 10 different settings, i.e., {100, 200,

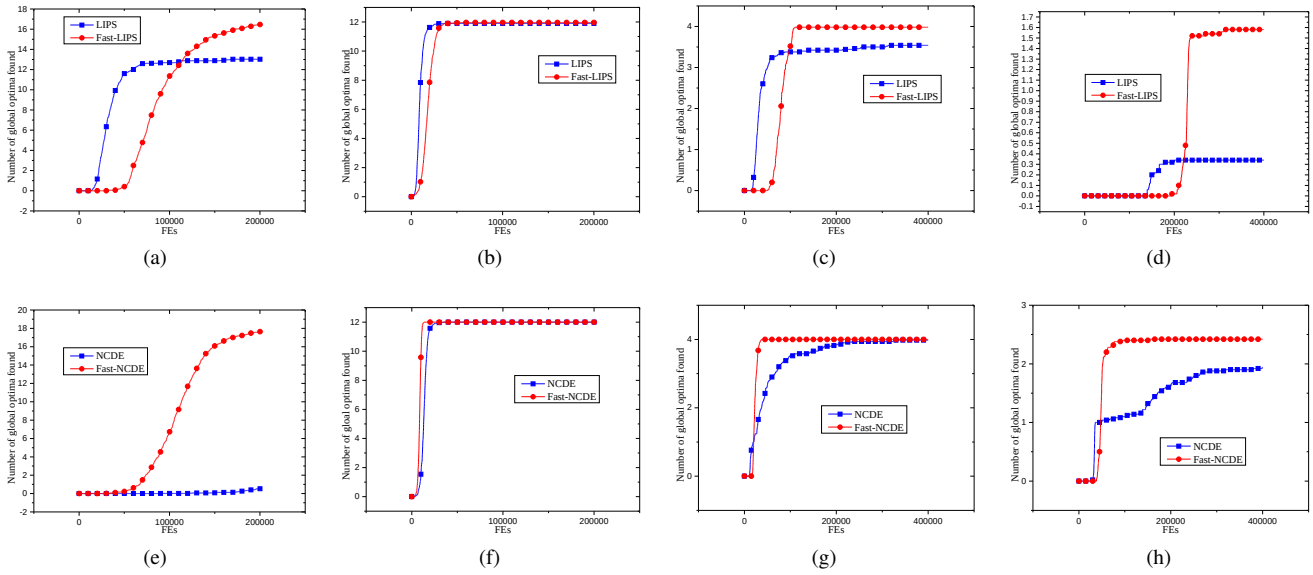

Fig. 7. Comparison of convergence speeds of LIPS, Fast-LIPS, NCDE, and Fast-NCDE. (a)-(d): LIPS versus Fast-LIPS on F6, F10, F14, and F18. (e)-(h): NCDE versus Fast-NCDE on F6, F10, F14, and F18.

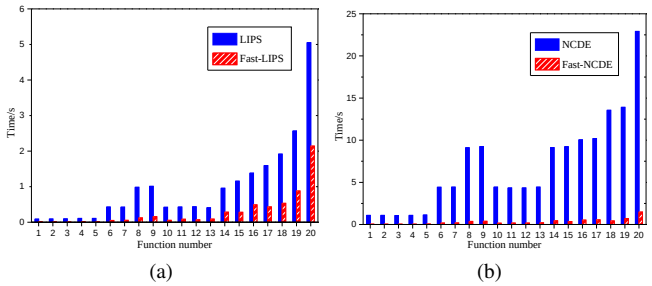

Fig. 8. Running time of LIPS, Fast-LIPS, NCDE, and Fast-NCDE on the 20 benchmark test functions. (a) LIPS versus Fast-LIPS. (b) NCDE versus Fast-NCDE.

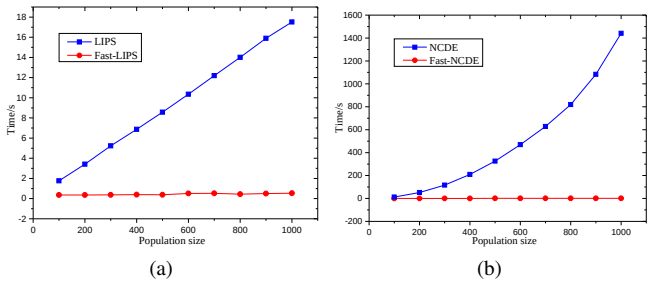

Fig. 9. Running time against the population size. (a) LIPS and Fast-LIPS. (b) NCDE and Fast-NCDE.

$\ldots, 1000\}$ are employed by the algorithms and are tested on F18. Experimental results are shown in Fig. 9, which plots the running time against the population size. It can be seen that the running time of Fast-LIPS and Fast-NCDE is much shorter than that of LIPS and NCDE in all the settings of N_p. The differences become more salient as the population size grows. Since the termination criterion of the algorithms is defined by the MaxFEs, the polylines in Fig. 9 actually reflect the number of elementary operators taken per fitness evaluation. Under this assessment, the time complexity of the fast niching based multimodal algorithms is $O(D)$. As expected, it can be seen that the polylines representing Fast-LIPS and Fast-NCDE are nearly horizontal.

D. Integration with Other Algorithms

To test the efficiency of the fast niching technique, we integrate it within five other algorithms compared in the study, namely, DE/nrand/1, NSDE, NShDE, rpso (r3pso-lsh), and FER-PSO. We add the prefix "Fast-" to the algorithms to denote the variants. For Fast-DE/nrand/1, the nearest neighbor of a target vector is found in the bucket to which the target vector is hashed. For Fast-NSDE, after a species seed has been detected, other members of the species are selected from the same bucket as the species seed. For Fast-NShDE, when generating a donor vector $\boldsymbol{v}_i$, the vectors $\boldsymbol{x}_{r1}$, $\boldsymbol{x}_{r2}$, and $\boldsymbol{x}_{r3}$ are chosen from the bucket in which the target vector $\boldsymbol{x}_i$ locates. For Fast-rpso, the ring topology of particles is defined within each bucket. For Fast-FER-PSO, the "fittest-and-closest" neighbor of a particle is detected within a bucket instead of from the entire swarm. The fast niching versions of CDE, SDE, ShDE, and DE/inrand/1 will be the same as those of NCDE, NSDE, NShDE, and DE/nrand/1. Experiments have been conducted on the benchmark functions to investigate the performance of the integrated algorithms. The results are summarized in Tables S5-S9. Table S10 gives the results of the multiple-problem Wilcoxon's test. In addition, the running time of the algorithms is plotted in Fig. S1. From the results, it can be observed that the fast niching versions of the algorithms are able to achieve similar or better performance using equal or less running time.

E. Performance on High Dimensional Multimodal Problems

In this part, we carry out experiments to compare the performance of the niching algorithms on very high dimensional

problems. Specifically, the Modified Rastrigin function and two composition functions (CF1 and CF2) are used to test the algorithms' scalability. For the Modified Rastrigin function, seven different settings of the number of dimensions, i.e., 50, 100, 200, 400, 600, 800, and 1000, are adopted. The maximum number of fitness evaluations is fixed at $5000D$ and the number of global optima is set to 48. For CF1 and CF2, since no algorithm can find more than one optimum when the number of dimensions is larger than 200, the following settings of the number of dimensions are used instead: 10, 20, 40, 60, 80, 100, and 200. The maximum number of fitness evaluations for the two functions is set to $10{,}000D$. CF1 has six global optima while CF2 has eight. The population size of the algorithms is set as $N_p = 200$. The PR values (at the accuracy level 1E$-$04) are summarized in Tables S11-S13. Only algorithms that can produce not all zero results are reported in the tables. From the experimental results, it can be seen that the LSH-based niching technique can work well in high dimensional search space. The performance of Fast-NCDE remains very competitive under different numbers of dimensions. In comparison, the performance of the other algorithms deteriorates significantly as the number of dimensions increases. This may due to the fact that in high dimensional space, not only the exploitation ability, but also the exploration ability, are crucial to the detection of multiple optima.

F. Investigation of Parameter Settings

1) Effect of the Number of Buckets: We continue to investigate the effect of parameters introduced by the fast niching technique. The first parameter nb is used to determine the number of buckets and it controls the parameter r of the hash functions in an indirect way. A suitable value of nb will help induce stable niching behavior. The effect of nb on the performance of Fast-LIPS and Fast-NCDE is tested by using six different values: 5, 10, 15, 20, 25, and 30. For the sake of clarity, Fast-NCDE with the value k is denoted as Fast-NCDE-nb$k$. This naming rule also applies to Fast-LIPS. The PR values of the algorithms at the accuracy level $\varepsilon =$1E$-$04 are summarized in Tables S14-S15. The best results are in bold. In addition, rankings of the algorithms obtained through the Friedman's test are given in Table VIII. We can observe that for Fast-LIPS, the setting $nb = 30$ provides the best performance, while for Fast-NCDE, $nb = 5$ is preferred. The difference comes from the different reproduction mechanisms of PSO and DE. In Fast-LIPS, when updating the velocity of a particle $\boldsymbol{x}$, several members in the same bucket as $\boldsymbol{x}$ are selected and are combined to form an exemplar. The exemplar has a direct influence on the moving trajectory of $\boldsymbol{x}$. If particles from several regions of attractions are in the same bucket, $\boldsymbol{x}$ might oscillate between different peaks. The oscillation can be eliminated by reducing the bucket size r. This suggests the setting of a large value for nb. Unlike Fast-LIPS, in Fast-NCDE, members in the same bucket as $\boldsymbol{x}$ are used to generate donor vectors. Diverse donor vectors can be produced by using the individuals that are spaced from each other. This suggests the setting of a small value for nb.

To study the influence of the increasing dimensionality on the setting of nb, experiments are conducted on the scalable Modified Rastrigin function using seven different numbers of dimensions (i.e., 50, 100, 200, 400, 600, 800, and 1000). Experimental results of Fast-NCDE are provided in Table S16. In addition, the convergence speed of the algorithm is given in Table S17. From the tables, it can be seen that the setting $nb = 5$ is able to achieve the overall best performance on problems with different scales. This outcome is in line with the rankings shown in Table VIII, which indicates that the best setting of nb does not depend on the problem dimension. As shown in Table S16, the PR values obtained by Fast-NCDE-nb5 are consistently larger than those of other algorithms when the number of dimensions is greater than 100. For the 50-dimensional problem, the convergence speed of Fast-NCDE-nb5 is slightly slower than the algorithms with larger values of nb, as reported in Table S17. To illustrate, Fig. 10 plots the convergence speed of Fast-NCDE-nb5 and Fast-NCDE-nb25 on the 50- and 400-dimensional problems. It can be observed that algorithms with small values of nb can generally detect more optima, while algorithms with large values of nb are able to quickly converge to a certain number of optima. According to the theoretical analysis of the fast niching technique, the parameter nb is able to adjust the ratio of exploitation and exploration. A small value of nb can increase the ratio of exploration while a larger value can increase the ratio of exploitation. This agrees with the results reported in Table S16 and Fig. 10. Generally, to handle complex or high (100 or more) dimensional problems, we need to increase the exploration ability. A small value of nb (around 5) would be appropriate for such scenarios. On the other hand, for low dimensional simple problems, a large value of nb (around 25) can be used to accelerate the convergence speed.

2) Effect of the Number of Hash Functions: We finally study the effect of the parameter nh. nh controls the number of hash functions sampled from the LSH family. By using multiple hash functions, it is expected that close individuals will be hashed into the same bucket by at least one of the hash functions. In this way, we can have a guarantee on the local

TABLE VIII
RANKING OF FAST NICHING MULTIMODAL ALGORITHMS WITH DIFFERENT SETTINGS OF *NB* BY THE FRIEDMAN'S TEST

Algorithm	Ranking	Algorithm	Ranking
Fast-LIPS-nb30	2.425	Fast-NCDE-nb5	2.3
Fast-LIPS-nb25	2.575	Fast-NCDE-nb10	2.825
Fast-LIPS-nb20	3.225	Fast-NCDE-nb20	3.5
Fast-LIPS-nb15	3.675	Fast-NCDE-nb15	3.825
Fast-LIPS-nb10	4.2	Fast-NCDE-nb25	4.05
Fast-LIPS-nb5	4.9	Fast-NCDE-nb30	4.5

TABLE IX
RANKING OF FAST NICHING MULTIMODAL ALGORITHMS WITH DIFFERENT SETTINGS OF *NH* BY THE FRIEDMAN'S TEST

Algorithm	Ranking	Algorithm	Ranking
Fast-LIPS-nh10	3.1	Fast-NCDE-nh20	3.075
Fast-LIPS-nh15	3.45	Fast-NCDE-nh25	3.3
Fast-LIPS-nh25	3.45	Fast-NCDE-nh15	3.4
Fast-LIPS-nh5	3.575	Fast-NCDE-nh30	3.525
Fast-LIPS-nh20	3.7	Fast-NCDE-nh5	3.725
Fast-LIPS-nh30	3.725	Fast-NCDE-nh10	3.975

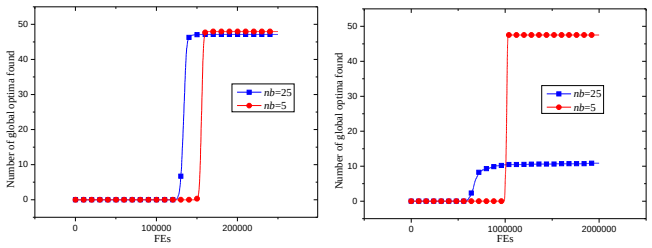

Fig. 10. Comparison of convergence speeds of Fast-NCDE-nb5 and Fast-NCDE-nb25. (a) $50D$. (b) $400D$.

search and fine-tuning abilities of EAs. To ascertain the effect of the parameter, we tested six different settings: 5, 10, 15, 20, 25, and 30. Fast-LIPS with the above settings are named in the form of Fast-LIPS-nh$k$. By analogy, Fast-NCDE with different nh values have their names like Fast-NCDE-nh$k$. The PR values of the algorithms at the accuracy level ε =1E−04 are listed in Tables S18-S19. The best results are marked in bold. As shown in the table, no one algorithm is best for all the test functions. The difference between the PR values for each test function is small. Considering the statistical test results in Table IX, we can see that the algorithms with different nh values have similar rankings. This is observed for both Fast-LIPS and Fast-NCDE, which implies that the setting of nh is independent of the selection of the baseline algorithm.

The effect of nh is also tested on the scalable Modified Rastrigin function. The number of dimensions varies from 50 to 1000. Table S20 tabulates the PR values (at the accuracy level 1E−04) obtained by Fast-NCDE with different nh values. From the table, it can be seen that the PR values of Fast-NCDE-nh5 are slightly worse than those of the other five algorithms when the number of dimensions is set to 600 or more. This might suggest that the required number of hash functions will increase slightly as the number of dimensions grows. Despite this, the difference between the algorithms with nh ranges from 15 to 25 is insignificant. The range is consistent with the results shown in Table IX. Overall, the performance of the fast niching algorithms is not very sensitive to the setting of nh. According to the experimental results, a value between 15 and 25 is able to provide very stable performance for problems with different scales.

VI. Conclusion

In this paper, a fast niching technique is developed to reduce the computational effort of evolutionary multimodal algorithms. Individuals in the population are projected to a number of buckets using efficient locality sensitive hash (LSH) functions. We show that stable niching behavior can be induced by limiting the interaction of individuals to each bucket. Theoretical analysis performed in the paper suggests that the LSH method greatly reduces the computational overhead of existing niching techniques, and it can preserve the time complexity of basic EAs, i.e., $O(D \cdot N_p)$. Moreover, the strategy has an intrinsic property that it keeps a balance between the exploration and exploitation abilities of multimodal algorithms. We integrate the proposed niching technique into two state-of-the-art multimodal algorithms, LIPS and NCDE, and propose Fast-LIPS and Fast-NCDE. The proposed algorithms have been evaluated on a set of commonly used multimodal benchmark functions. The experimental results demonstrate that the fast niching-based multimodal algorithms are not only very fast, but also very competitive when compared with the state-of-the-art algorithms. It would be an interesting future research direction to incorporate the fast niching technique within more multimodal algorithms. Moreover, by incorporating discrete LSH families, it is now possible to develop efficient discrete niching techniques for solving discrete multimodal problems.

References

[1] E. Dilettoso and N. Salerno, "A self-adaptive niching genetic algorithm for multimodal optimization of electromagnetic devices," *IEEE Trans. Magn*, vol. 42, no. 4, pp. 1203–1206, Apr. 2006.
[2] E. Perez, F. Herrera, and C. Hernandez, "Finding multiple solutions in job shop scheduling by niching genetic algorithms," *J. Intell. Manuf.*, vol. 14, no. 3-4, pp. 323–339, Jun. 2003.
[3] M. Boughanem and L. Tamine, "A study on using genetic niching for query optimization in document retrieval," in *Proc. Adv. Inform. Retrieval*, 2002, pp. 93–100.
[4] G. Delvecchio, C. Lofrumento, F. Neri, and M. S. Labini, "A fast evolutionary-deterministic algorithm to study multimodal current fields under safety level constraints," *Int. J. Comput. Math. Electr. Electron. Eng.*, vol. 25, no. 3, pp. 599–608, 2006.
[5] J. Yao, N. Kharma, and P. Grogono, "Multipopulation genetic algorithm for robust and fast ellipse detection," *Pattern Anal. Applicat.*, vol. 8, no. 1, pp. 149–162, 2005.
[6] T. Back, D. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*. Oxford, U.K.: Oxford Univ. Press, 1997.
[7] D. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning*. Boston: Addison Wesley, 1989.
[8] L. Tang, Y. Dong, and J. Liu, "Differential evolution with an individual-dependent mechanism," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 560–574, Aug 2015.
[9] H. Wu, C. Nie, F. C. Kuo, H. Leung, and C. J. Colbourn, "A discrete particle swarm optimization for covering array generation," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 575–591, Aug 2015.
[10] S. W. Mahfoud, "Niching methods for genetic algorithms," Ph.D. dissertation, Dept. Comput. Sci., Univ. Illinois Urbana-Champaign, Urbana, 1995.
[11] B. Sareni and L. Krahenbuhl, "Fitness sharing and niching methods revisited," *IEEE Trans. Evol. Comput.*, vol. 2, no. 3, pp. 97–106, Sep. 1998.
[12] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proc. 2nd Int. Conf. Genet. Algorithms*, 1987, pp. 41–49.
[13] K. D. Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, University of Michigan, United States, 1975.
[14] G. R. Harik, "Finding multimodal solutions using restricted tournament selection," in *Proc. 6th Int. Conf. Genet. Algorithms*, 1995, pp. 24–31.
[15] J. P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson, "A species conserving genetic algorithm for multimodal function optimization," *Evol. Comput.*, vol. 10, no. 3, pp. 207–234, 2002.
[16] R. Thomsen, "Multimodal optimization using crowding-based differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2004, pp. 1382–1389.
[17] X. Li, "Efficient differential evolution using speciation for multimodal function optimization," in *Proc. Conf. Genet. Evol. Comput.*, 2005, pp. 873–880.
[18] D. Parrott and X. Li, "Locating and tracking multiple dynamic optima by a particle swarm model using speciation," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 440–458, Aug. 2006.
[19] B. Y. Qu, P. N. Suganthan, and J. J. Liang, "Differential evolution with neighborhood mutation for multimodal optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 5, pp. 601–614, Oct. 2012.
[20] B. Y. Qu, P. N. Suganthan, and S. Das, "A distance-based locally informed particle swarm model for multimodal optimization," *IEEE Trans. Evol. Comput.*, vol. 17, no. 3, pp. 387–402, Jun. 2013.

[21] S. Biswas, S. Kundu, and S. Das, "Inducing niching behaviour in differential evolution through local information sharing," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 246–263, Apr. 2015.
[22] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimension via hashing," in *Proc. 25th Int'l Conf. Very Large Data Bases*, 1999, pp. 518–529.
[23] P. Indyk and R. Motwani, "Approximate nearest neighbor: Towards removing the curse of dimensionality," in *Proc. 30th Ann. ACM Symp. Theory of Computing*, 1998, pp. 604–613.
[24] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 1991, pp. 69–93.
[25] A. Basak, S. Das, and K. C. Tan, "Multimodal optimization using a biobjective differential evolution algorithm enhanced with mean distance based selection," *IEEE Trans. Evol. Comput.*, vol. 17, no. 5, pp. 666–685, Oct. 2013.
[26] W. Song, Y. Wang, H.-X. Li, and Z. Cai, "Locating multiple optimal solutions of nonlinear equation systems based on multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 3, pp. 414–431, 2015.
[27] J. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.
[28] X. Li, "Niching without niching parameters: Particle swarm optimization using a ring topology," *IEEE Trans. Evol. Comput.*, vol. 14, no. 1, pp. 150–169, Feb. 2010.
[29] A. Petrowski, "A clearing procedure as a niching method for genetic algorithms," in *Proc. IEEE Int. Conf. Evol. Comput.*, May 1996, pp. 798–803.
[30] X. Li, "A multimodal particle swarm optimizer based on fitness euclidean-distance ration," in *Proc. Genet. Evol. Computat. Conf.*, 2007, pp. 78–85.
[31] T. Otani, R. Suzuki, and T. Arita, "DE/isolated/1: a new mutation operator for multimodal optimization with differential evolution," *International Journal of Machine Learning and Cybernetics*, vol. 4, no. 2, pp. 99–105, 2012.
[32] M. G. Epitropakis, V. P. Plagianakos, and M. N. Vrahatis, "Finding multiple global optima exploiting differential evolutions niching capability," in *IEEE Symp. on Differential Evolution (SDE)*, Apr. 2011, pp. 1–8.
[33] ——, "Multimodal optimization using niching differential evolution with index-based neighborhoods," in *Proc. IEEE Int. Conf. Evol. Comput.*, Jun. 2012, pp. 1–8.
[34] S. Biswas, S. Kundu, and S. Das, "An improved parent-centric mutation with normalized neighborhoods for inducing niching behavior in differential evolution," *IEEE Trans. Cybern.*, vol. 44, no. 10, pp. 1726–1737, Oct. 2014.
[35] L. Li and K. Tang, "History-based topological speciation for multimodal optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 1, pp. 136–150, Feb 2015.
[36] R. Brits, A. Engelbrecht, and F. van den Bergh, "A niching particle swarm optimizer," in *Proc. 4th Asia-Pacific Conf. SEAL*, 2002, pp. 692–696.
[37] R. Ursem, "Multinational evolutionary algorithms," in *Proc. IEEE Congr. Evol. Comput.*, 1999, pp. 1633–1640.
[38] B. Rigling and F. Moore, "Exploitation of subpopulations in evolutionary strategies for improved numerical optimization," in *Proc. 11th Midwest Artif. Intell. Cogn. Sci. Conf.*, 1999, pp. 80–88.
[39] D. Zaharie, "A multipopulation differential evolution algorithm for multimodal optimization," in *Proc. 10th Mendel Int. Conf. Soft Comput.*, Jun. 2004, pp. 17–22.
[40] W. Gao, G. G. Yen, and S. Liu, "A cluster-based differential evolution with self-adaptive strategy for multimodal optimization," *IEEE Trans. Cybern.*, vol. 44, no. 8, pp. 1314–1327, Aug. 2014.
[41] L. Qing, W. Gang, Y. Zaiyue, and W. Qiuping, "Crowding clustering genetic algorithm for multimodal function optimization," *Applied Soft Computing*, vol. 8, no. 1, pp. 88–95, 2008.
[42] G. Zhang, L. Yu, Q. Shao, and Y. Feng, "A clustering based GA for multimodal optimization in uneven search space," in *Proc. 6th World Congress on Intelligent Control and Automation*, 2006, pp. 3134–3138.
[43] F. Streichert, G. Stein, H. Ulmer, and A. Zell, "A clustering based niching EA for multimodal search spaces," in *Artificial Evolution*. Springer, 2003, pp. 293–304.
[44] A. Passaro and A. Starita, "Particle swarm optimization for multimodal functions: A clustering approach," *J. Artif. Evol. App.*, vol. 2008, pp. 1–15, Jan 2008.
[45] A. El Imrani, A. Bouroumi, H. Z. El Abidine, M. Limouri, and A. Essaıd, "A fuzzy clustering-based niching approach to multimodal function optimization," *Cognitive Systems Research*, vol. 1, no. 2, pp. 119–133, 2000.
[46] A. Rajaraman and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press, 2011.
[47] S. Hui and P. N. Suganthan, "Ensemble and arithmetic recombination-based speciation differential evolution for multimodal optimization," *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 64–74, Jan 2016.
[48] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, "Locality sensitive hashing scheme based on p-stable distributions," in *Proc. 20th Symp. Computational Geometry*, 2004, pp. 253–262.
[49] X. Li, A. Engelbrecht, and M. G. Epitropakis, "Benchmark functions for CEC'2013 special session and competition on niching methods for multimodal function optimization," Evol. Comput. Mach. Learn. Group, RMIT University, Melbourne, VIC, Australia, Tech. Rep., 2013.

Yu-Hui Zhang received the B.S. degree from Sun Yat-sen University, China, in 2013. Currently, he is a research assistant with School of Computer Science and Engineering, South China University of Technology, while he is also a Ph.D. candidate with School of Data and Computer Science, Sun Yat-sen University, China.

His research interests include evolutionary computation, swarm intelligence, multimodal optimization, and their real-world applications.

Yue-Jiao Gong (S'10-M'15) received the B.S. and Ph.D. degree in Computer Science from Sun Yat-sen University, China in 2010 and 2014, respectively. She is currently a research fellow with School of Computer Science and Engineering, South China University of Technology, China.

Her research interests include evolutionary computation, swarm intelligence, and their applications to intelligent transportation scheduling, and image processing.

Hua-Xiang Zhang is currently a professor with Shandong Normal University, China. He received his Ph.D. from Shanghai Jiaotong University in 2004, and worked as an associated professor with the Department of Computer Science, Shandong Normal University from 2004 to 2005. He has authored over 100 journal and conference papers and has been granted 8 invention patents. His current research interests include machine learning, pattern recognition, evolutionary computation, web information processing, etc.

Tian-Long Gu received the M.Eng. degree from Xidian University, China, in 1987, and the Ph.D. degree from Zhejiang University, China, in 1996. From 1998 to 2002, he was a Research Fellow with the School of Electrical and Computer Engineering, Curtin University of Technology, Australia, and a Post-Doctoral Fellow with the School of Engineering, Murdoch University, Australia. He is currently a Professor with Guilin University of Electronic Technology, China. His research interests include formal methods, data and knowledge engineering, software engineering, and information security protocol.

Jun Zhang (M'02-SM'08) received the Ph.D. degree from the City University of Hong Kong, Hong Kong, in 2002. He is currently a Changjiang Chair Professor with School of Computer Science and Engineering, South China University of Technology, China.

His research interests include computational intelligence, cloud computing, data mining, and power electronic circuits. He has published over 200 technical papers in his research area. Dr. Zhang was a recipient of the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011 and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE Trans. on Evolutionary Computation, IEEE Trans. on Industrial Electronics and IEEE Trans. on Cybernetics.