A Close Neighbor Mobility Method Using Particle Swarm Optimizer for Solving Multimodal Optimization Problems

Juan Zou[a,b], Qi Deng[a,b,*], Jinhua Zheng[a,b,c], Shengxiang Yang[a,d,*]

[a] *Key Laboratory of Intelligent Computing and Information Processing, Ministry of Education, Information Engineering College of Xiangtan University, Xiangtan, Hunan Province, China*
[b] *Faculty of Informational Engineering University of Xiangtan, Xiangtan, 411105, China*
[c] *Hunan Provincial Key Laboratory of Intelligent Information Processing and Application, Hengyang, 421002, China*
[d] *School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH,U.K.*

Abstract

Niching is an important technique for multimodal optimization. Most existing niching methods require specification of certain niching parameters in order to perform well. But these parameters are usually difficult to set because they depend on the problem. The particle swarm optimization algorithm using the ring neighborhood topology does not require any niche parameters, but the determination of the particle neighborhood in this method is based on the subscript of the particle, and the result fails to achieve the best performance. For better performance, in this paper, a particle swarm optimization algorithm based on the ring neighborhood topology of Euclidean distance between particles is proposed, which is called the close neighbor mobility optimization algorithm. The algorithm mainly includes the following three strategies: elite selection mechanism, close neighbor mobility strategy and modified DE strategy. It mainly uses the Euclidean distance between particles. Each particle forms its own unique niche, evolves in a local scope, and finally locates multiple global optimal solutions with high precision. The algorithm greatly improves the accuracy

*Corresponding author: Qi Deng, Shengxiang Yang
Email addresses: 564049349@qq.com (Qi Deng), syang@dmu.ac.uk (Shengxiang Yang)

of the particle. The experimental results show that the close neighbor mobility optimization algorithm has better performance than most single-objective multi-modal algorithms.

Keywords: Multimodal optimization, particle swarm optimization, ring neighborhood topology, niche.

1. Introduction

There are many optimization problems that need as many global optimal solutions as possible in the real world, such as in a power system [1], protein structure prediction [2], and data mining [3, 4]. These problems are commonly known as multimodal optimization problems (MMOPs). There are two aspects that are important in the optimization of multimodal problems. The first is that for most multimodal optimization algorithms, if the required accuracy is high, even if the final solution obtained by the algorithm is close to the real peak (In multimode optimization, the peak represents the optimal value of function optimization), it is difficult for these solutions to reach the exact position of the peak. In the second, many algorithms have difficulty exploring all regions, and some of the highest peaks are easily missed. Especially for some spikes, the decision space occupied by peaks is very small, and it is difficult for particles to explore these peaks. Many existing algorithms try to balance the number of global optimal solutions and the accuracy of global optimal solutions, but it is difficult to ensure that both are high [5].

The basic idea of the niche method comes from the fact that organisms always live together with their own species in the process of evolution. It is reflected in the evolution algorithm that individuals in the evolution algorithm evolve in a specific living environment. Compared with other commonly used optimization methods, it has high search efficiency and can get multiple extremum points of objective function in one search, which is suitable for solving MMOPs. In the field of evolutionary computation, there has been a growing interest in applying evolutionary algorithms (EAs) to solve MMOPs. For exam-

ple, classification problems in machine learning can be mapped to MMOPs and hence be treated by an EA employing a niching method [6]. A niching genetic algorithm (GA) was also applied to the problem of the inversion of teleseismic waves [7]. For multimode problems, the most critical part is to find as many global optimal solutions as possible and improve the accuracy of the solutions as much as possible. In evolutionary multiobjective optimization, because of the advantages of niche methods, niching methods are often used to maintain solution diversity[8].

The concept of neighborhood has been widely used in EAs. In general, neighborhood relationships can be divided into two broad categories, namely index-based and distance-based. In a single global peak optimization scheme, index-based neighborhoods are typically used, especially in Particle Swarm Optimization (PSO). Different topological-based PSO algorithms have been proposed and compared. These topologies are based on population subscripts (index-based). The advantage of this is that it can save computing resources. In 2004, Mendes [9] proposed a fully-informed PSO that also uses topological and index-based neighborhoods as the basic structure. One of the earliest topological index-based neighborhood differential evolution (DE) works was carried out by Tasoulis [10]. The algorithm divides the population into different sub-populations and uses a ring topology to exchange information between different sub-populations. This method was modified and further improved by Weber *et al.*[11] [12]. Das *et al.* used the index-based neighborhood concept of each population member to improve the performance of DE [13][14].

Most of the same types of algorithms are index-based, but the accuracy of the solution is not the optimal value in most cases. The PSO algorithm, which imitates the foraging behavior of bird flocks, is a very effective method for multimodal problems. PSO is popular because it is easy to implement, and has strong optimization ability. Its efficiency in solving complex optimization problems has attracted significant research [15]. In order to improve the accuracy of the solution, a PSO algorithm based on the ring neighborhood topology of Euclidean distance between particles is proposed in this paper, which is called

the close neighbor mobility optimization algorithm (CNMM). The algorithm greatly improves the accuracy of the particle. The algorithm mainly includes the following three strategies: elite selection mechanism, close neighbor mobility strategy and modified DE strategy. It mainly uses the Euclidean distance between particles. Each particle forms its own unique niche, evolves in a local scope, and finally locates multiple global optimal solutions with high precision.

The reminder of this paper is organized as follows. Section 2 reviews the Particle Swarm Optimizer (PSO) and the basic DE. Section 3 gives a detailed description of the proposed CNMM algorithm. Experimental setup and results are presented and compared in Section 4.The experimental results show that the CNMM algorithm has better performance than most single-objective multimodal algorithms. Finally the paper is concluded in Section 5.

2. FUNDAMENTAL KNOWLEDGE

Since the principles of PSO and DE [16, 17, 18] are used in the CNMM algorithm, we first introduce these two classic algorithms, which will facilitate the detailed description of CNMM.

A.Particle swarm optimization algorithm

PSO [19, 20, 21] searches by simulating group behaviors such as those of flocks, fish, and herds. PSO has many advantages, so it has a wide range of applications in dealing with multimodal problems [22].

PSO is based on intelligent algorithms of populations. Each individual in the population is called a particle, and each particle represents a solution. The purpose of each particle is to constantly approach the location of the food, and assume that the position of the food is a global optimal solution. To achieve this, each particle ultimately approaches the location of the food by using the best position (pbest) that it has experienced and the best position (gbest) of all the particles in the population.

The mathematical description of the PSO algorithm is as follows. Sup-

pose that the dimension of each particle's decision variable in the population is n; the size of the population is $N(i = 1, 2, ..., N)$; the position vector and velocity vector of each particle are represented as $x_i = (x_{i1}, x_{i2}, ..., x_{in})$ and $v_i = (v_{i1}, v_{i2}, ..., v_{in})$, respectively, and the particle i is in $d(d = 1, 2, ..., n)$ dimension. The speed and position updates from time t to time $t + 1$ are as follows:

$$\begin{aligned} v_{i,d} = & \ w * v_{i,d}(t) + c_1 * r_1 * (pbest_{id}(t) - x_{i,d}(t)) \\ & + c_2 * r_2 * (gbest_{id}(t) - x_{i,d}(t)). \end{aligned} \tag{1}$$

$$x_{id}(t + 1) = v_{id}(t) + x_{id}(t), \tag{2}$$

where w is the inertia weight, indicating how much of the original velocity of the particle can be retained. Larger w equals strong global search ability and weak local search ability. Small w equals strong local search ability and weak global search ability. C_1 and C_2 are the individual's own learning and social learning factors, respectively. r_1 and r_2 are random numbers of $[0, 1]$. It is the d-dimensional component of the best position of the particle i, which is the d-th dimension component of the best position of the group.

The principle of the PSO algorithm is shown in Fig.1. In Fig.1, the X_t represents the initial position of a particle at time t; pbest represents the best position in particle X_t history, and gbest represents the best position of all particles in the entire population. V_1 represents the guiding effect of pbest on the particles. V_2 represents the guiding effect of gbest on the particles, and V_3 represents the influence of the velocity of the particles at the last moment on the particles. The shape of this triangle represents the location of the food. In Fig.1, according to the law of parallelograms, under the combined influence of V_1, V_2 and V_3, x_t reaches a new position x_{t+1}, and x_{t+1} is closer to the position of the food. From Fig.1 we can clearly see that the core idea of the PSO algorithm is that each particle is continuously close to the global optimal position under the leading role of better particles.

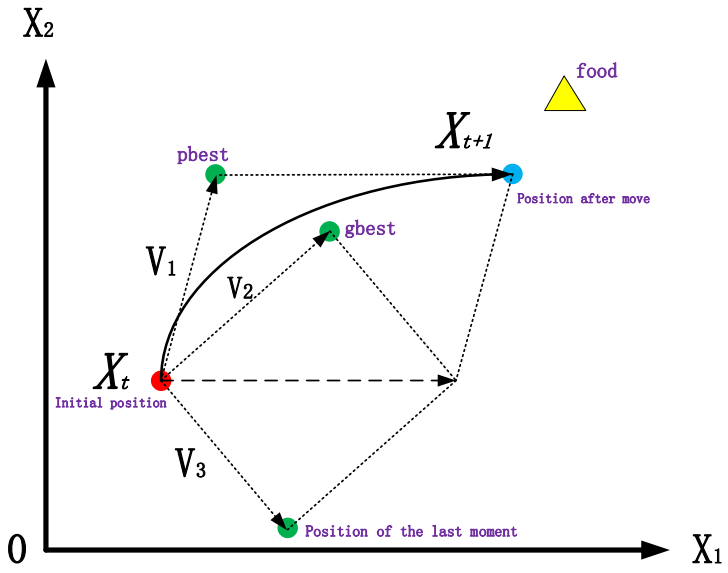


Figure 1: The principle of particle swarm optimization

B.Differential evolution(DE)

Storn and Price [21] introduced DE to solve unconstrained single-objective optimization problems. Due to the unique characteristics of DE, it is also widely used in multimodal problems [23]. The basic idea is to mutate and cross-operate the current population to produce a new population. Then the selection operation is used to make a one-to-one selection of the two populations to produce the final newly generated population. Specifically, there are variations, crossovers, and the selection of three operations. First, the mutation operation is performed using formula (3).

$$v_i = x_{r1} + F \times (x_{r2} - x_{r3}), \tag{3}$$

where $i = 1, ..., N$(N is the size of the population). The parameter v_i is the result of the i-th particle variation. The $r1, r2, r3$ are three particles randomly selected from the population($r1, r2, r3 \neq i$). Parameter F is a positive real con-

trol parameter called the amplification factor, which controls the amplification of difference vectors.

By using formula (4), v_i and x_i to perform the exchange operation on each dimension, a new individual $u_{i,j}$ is finally generated.

$$u_{i,j} = \begin{cases} v_{i,j} & if \leq CR \ or \ j = j_{rand} \\ x_{i,j} & otherwise, \end{cases} \tag{4}$$

where $j = 1, ..., D(D$ is the dimension of the particle decision variable); $rand_j$ is a uniformly distributed random number between 0 and 1 that is regenerated for each j; j_{rand} is an integer randomly chosen from $[1, .., D]$. CR is the crossover control parameter, and $u_{i,j}$ is the jth element of the trial vector u_i. The idea of this step is to exchange some content between v_i and x_i to form a new u_i.

Finally, the selection is performed between the target vector x_i and the newly generated vector u_i, and the better one will be saved to the next generation of the population.

$$x_i = \begin{cases} u_i & if \ f(u_i) \geq f(x_i) \\ x_i & otherwise, \end{cases} \tag{5}$$

where $f(x)$ is the fitness evaluation function for a particle x.

The principle of the modified DE operator is shown in Fig.2. The green dot X_{r0} represents the current particles. The black X_{r1}, blue X_{r3}, and red X_{r2} dots represent three particles randomly selected from the particle group ($X_{r0} \neq X_{r1} \neq X_{r2} \neq X_{r3}$). As described in Equation (3) of Section 2-B, the black point X_{r1} determines the starting point of the movement. The blue point X_{r3} and the red point X_{r2} determine the direction of movement, and parameter F determines the length of the movement. By Equation (3) of Section 2-B, X_{r1} moves to V_0. V_0 and the original point X_{r0} are cross-operated by Equation (4) of Section 2-B, as indicated by the open arrow in Fig.2. The green dot X_{r0} may move to V_0, V_1, V_2 or its original X_{r0} position.

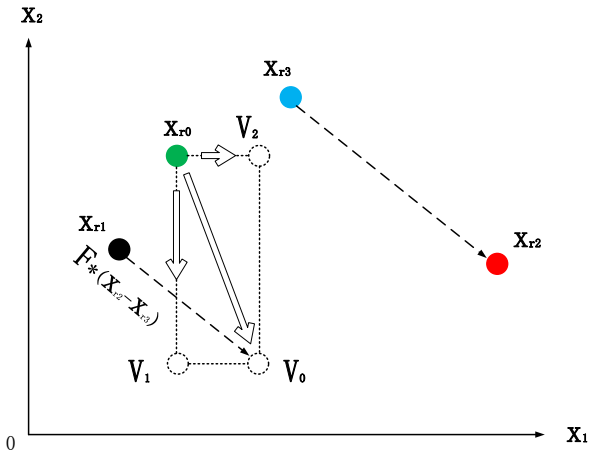


Figure 2: The principle of the DE method

3. Proposed Approach

Aiming at the problem that most single-objective multimodal algorithms solve the multimodal problem, the solution accuracy is not high. A PSO algorithm based on distance-based ring neighborhood topology is proposed, which is called CNMM.

In this section, we describe the details of the CNMM algorithm. The CNMM algorithm consists of three main strategies. The details of the elite selection strategy are described in Part A of Section 3. The details of the neighbor movement are described in Section 3-B. The details of the modified DE strategy are described in Section 3-C. The steps of the entire CNMM algorithm are described in Section 3-D. In the final Section 3-E, we simply express the working principle of CNMM through several diagrams.

A.Elite Selection Strategy

The aim of the elite selection strategy is to find the particles with good convergence and distribution in the whole population so that these particles

can guide the surrounding particles to explore. Algorithm 1 shows the detailed process of the elite selection strategy.

At the beginning of algorithm 1, the whole population is sorted in descending order according to fitness (Objective function value)(line 1). The most adaptable particle is found and put into the S set(line 3), which is used to store all elite particles. fb represents the fitness of the best particles in the current population(line 2).

Algorithm 1 :Elite selection mechanism

Input: The whole population P

Output: The collection S representing all elite particles collection of serial numbers

1: Arrange the whole population in descending order of fitness P_{sorted}
2: The highest fitness as fb
3: Particles with the highest fitness are added to the S set
4: **for** each particle $i \in P_{sorted}$ **do**
5: found $\leftarrow$ FALSE
6: **if** $fb - fit(i) \leq e$ **then**
7: found $\leftarrow$ TRUE
8: **for** each particle $s \in S$ **do**
9: **if** $Distance(s, i) \leq r$ **then**
10: found $\leftarrow$ FALSE;
11: break;
12: **end if**
13: **end for**
14: **end if**
15: **if** found==TRUE **then**
16: Let $S \leftarrow S \bigcup$ { particle i }
17: **end if**
18: **end for**

The convergence for each particle in the P_{sorted} population is determined.

The difference between fitness and *fb* is calculated (line 6). The difference represents the proximity to the current population peak. e is a threshold, which is used to represent a distance from the peak. If the proximity between them is within the specified threshold e, the convergence of the particle is better.

Then, the distribution of the particle is judged(lines 8-13). *Distance(s,i)* is the Euclidean distance between particle s and particle i. The S set stores all the elite particles, and if the current particle is not within the radius of all the particles in the S set, then the distribution of this particle is better. Finally, particle i with good current convergence and distribution is added to the S set(line 16). By judging all the particles in the population, we can find all the elite particles in the current population.

B. The Close Neighbor Mobility Strategy

The Close Neighbor Mobility Strategy was inspired by Yue [36]. The close neighbor mobility strategy is the core part of CNMM. Before introducing the close neighbor mobility strategy, we introduce the technology of niche. The basic idea of niche comes from the fact that in the process of evolution, organisms usually live together with their own species and reproduce together. To reflect this in the algorithm, we make the individuals in the algorithm evolve in a specific living environment. Because niche technology can form the advantages of multiple populations, the niche algorithm can avoid the large-scale propagation of individuals with high adaptive value in the later evolution period, and fill the whole population.

The close neighbor mobility strategy is inspired by the concept of species in niches. In nature, individuals in the same species exchange information in a specific environment. The specific method of the close neighbor mobility strategy is that each particle finds the best one of the three most recent particles in its own living environment, and the current particle continuously moves toward the best particles in the neighbor, thereby achieving the purpose of evolution.

Algorithm 2 shows the detailed process of the close neighbor mobility strategy. In Algorithm 2, for any particle i (line 2) in the population, the Euclidean

Algorithm 2 :Close Neighbor Mobility Strategy

Input: The whole population P and the number of neighbors each particle has is expressed as L

Output: The collection G (The set G of leading particles $gbest_i$ of each particle i)

1: Get the whole population P
2: **for** each particle $i \in P$ **do**
3: **for** each particle $j \in P$ $(j \neq i)$ **do**
4: $result[j] = Distance(i, j)$
5: **end for**
6: Sort result set in ascending order according to Euclidean distance
7: Find the L(In the experiment, L takes 3) particles closest to the current particle as a, b, c
8: Find the particle with the highest value of the objective function in a, b, c as the $gbest_i$ of the particle i.
9: **end for**
10: The $gbest_i$ of all particles constitutes a set G

distance of this particle from other particles in the population is calculated (lines 3-5). Then ,the three particles *a, b, c* (lines 6-7) closest to particle i are found. Finally, the particle with the largest value of the objective function in the particles *a, b, c* is found (line 8), and this particle is used as the gbest of particle i.

C.The modified DE strategy

The goal of multimodal problems is to find all global optimalities. Therefore, the largest area possible must be explored. In order to make the exploration area larger, in our own algorithm, the last step of the traditional DE strategy is removed. Thus, the new populations have a greater chance of being distributed more widely.

The modified DE strategy is as shown in Algorithm 3. For any particle i

Algorithm 3 :The modified DE strategy

Input: The whole population P,the collection S representing all elite particles collection of serial numbers

Output: Updated population P

1: **for** each particle $i \in P$ **do**
2: **if** $i \notin S$ **then**
3: Randomly select three particles not equal to i from the particle swarm, which are represented as x_1,x_2, x_3, respectively.
4: Generate v_i according to (3) of Section 2-B
5: Generate u_i according to (4) of Section 2-B
6: u_i instead of particle i
7: **end if**
8: **end for**

that does not belong to the collection S (line 2), three particles are randomly selected in the particle group (line 3), and a new particle u_i is generated by the formula (3) and (4) (lines, 4-5), and then the original particle i is replaced (line 6).

D.CNMM

There are three main strategies for the overall idea of the CNMM algorithm. The elite retention mechanism, close neighbor mobility strategy, and modified DE strategy. With the combination of the three strategies, the algorithm can find the global optimal solution.

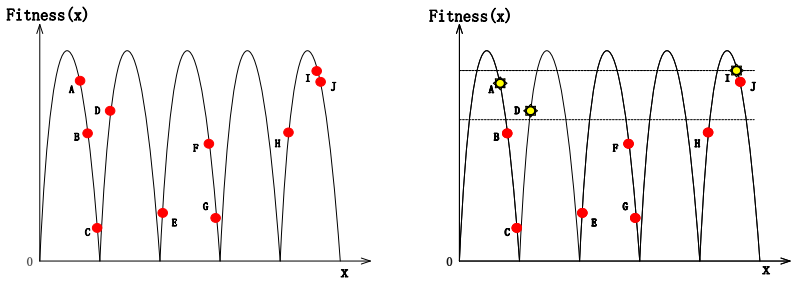
At the beginning of algorithm 3, the initial population is randomly generated, and each particle in the population has an initial position and velocity (line 1). Then the initial position of each particle is taken as the initial $pbest_i$ of the current particle i (line 2). Because, the algorithm does not reach the maximum number of iterations, the big loop is entered (lines 3-25). The excellent particles in the population in the S set are kept through the elite selection strategy of Algorithm 1 (line 4). Then the close neighbor mobility strategy of Algorithm 2

Algorithm 4 :CNMM

Randomly initialize the population P
Initialize $pbest_i$ of each particle i to itself, and get all particles $pbest_i$ to form set B
while Generation < MaxGenerations **do**
  Select elite individuals to update the S collection using Algorithm 1
  Execute Algorithm 2 to get G
  for i=1:ParticleNumber **do**
    if $i \in S$ **then**
      continue
    else
      Calculate the fitness of the current particle i
      if $fitness(i) > pbest_i$ **then**
        $pbest_i = particle\ i$
      end if
      Use the particle swarm algorithm to get the new position and velocity of the current *particle* i according to (1) and (2). ($pbest_i = B_i$,$gbest_i = G_i$)
    end if
    Check if all particle positions and velocities are within the specified range
    Evaluate the entire population
  end for
  if Generation *mod* MaxGeneration*K == 0 **then**
    if Generation < MaxGeneration **then**
      Select elite individuals to update the S collection using Algorithm 1
      In addition to the particles in the S collection, update the population using the modified DE strategy
    end if
  end if
  Generation=Generation+1
end while
Output the entire population P

is performed to obtain a set G of $gbest_i$ of each particle (line 5).

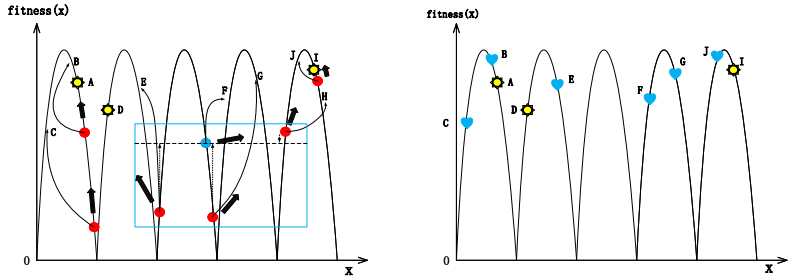
Next, the loop for each particle update is entered (lines 6-18). If particle i is an elite particle, then no update operation is performed (lines 7-8). Otherwise, after calculating the target function value of the current particle i, the current particle's $pbest_i$ is updated (lines 10-13). A very important step is to get the new position and velocity of the current particle i through the formulas (1) and (2) of the particle swarm (line 14). Note that in the particle swarm formula, the set B and the set G record the pbest and gbest of each particle respectively ($pbest_i = B_i, gbest_i = G_i$). The data are checked after the iteration of all particles in the population for correctness (line 16). Then the fitness of all particles is recalculated (line 17). After all particle iterations are completed, the specified number of iterations is reached, and the modified DE strategy is used to update the entire population (lines 19-23).



(a) (b)

Figure 3: The principle of the elite selection mechanism

E.Principles of CNMM Now we employ an example to illustrate the principles of CNMM through several pictures.



(a) (b)

Figure 4: The principle of the close neighbor mobility strategy

In the elite selection mechanism, as shown in Fig.3(a), we assume that there are 10 particles in the initial population. Through the operation of Algorithm 1, we get Fig.3(b). In Fig.3(b), the circle represents all particles and the small sun represents the elite particles found. Particles a, b, i, and j are all particles with higher fitness, but the distance between particle j and particle i is too close, so the final elite particles are a, b, j (The detailed process in Algorithm 1). Elite particles a, b and j benefit the overall algorithm in two ways. The first benefit is to ensure that the entire population does not degenerate. The second benefit is to guide the surrounding particles.

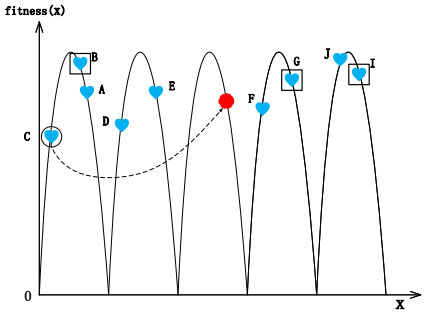


Figure 5: The effect of the modified DE operator

In the close neighbor mobility strategy, each particle starts looking for its own leader particle. In Fig.4(a), the little sun indicates that the elite parti-

cles do not move, and the small red circle indicates the particles that do. We take particle f as an example (a small blue circle), first finding the three particles e, g, j closest to the Euclidean distance of particle f in the decision space. The particles in the square are the three closest to particle f. Then, the particle with the largest objective function value among e, g, and j is used as the lead particle. The black bold arrow indicates the direction of movement of the current particle f at the next moment. Fig.4(b) indicates the population after the move. Particles that are not around the elite particles can move toward places where there may be peaks due to the guidance of excellent surrounding particles.

The reason why the close neighbor mobility strategy can continue to approach different peaks is because each particle with its three surrounding particles form a special niche. The current particle can quickly approach the nearest mountain. Additionally because the speed of particle movement in a particle swarm is limited, the most efficient way to move is to move toward the nearest mountain to avoid erroneous movement.

In Fig.4(b), no particles are present on the third peak. In order to prevent this phenomenon, after the evolution of a certain algebra, the DE strategy is used to update the entire population. When updating the population, the elite particles are not updated to prevent degradation after the entire population is updated. We update particles with poor target function values. As shown in Fig.5, when particle c is updated, three particles different from the current particle are randomly selected to regenerate the individual. The particles in the circle represent the current particles being updated. The particles in the square represent three randomly selected particles. Small circles represent newly generated particles. The new particles produced are likely to be distributed on the peaks without particles.

Fig.6 shows the results of a simple experiment of CNMM. In Fig.6(a), the initial population is generated. After eight generations as shown in Fig.6(b), the other particles in the population have basically found all the global optimal solutions. After eight generations of particles, almost all of the particles have

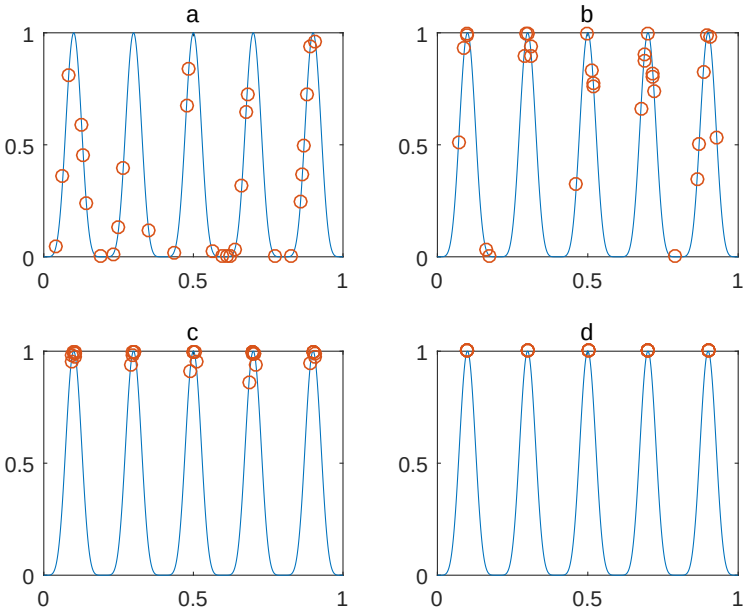


Figure 6: A simple example of CNMM

converged to the vicinity of the optimal solution (Fig.6(c)). In Fig.6(d), we see that all particles converge to the nearest peak. The process of Fig.6 simply demonstrates the validity of the CNMM principle.

4. Experimental Study

In this section, a series of experiments is used to verify the efficiency and feasibility of the CNMM algorithm. First, the benchmark function and the parameter settings in the experiment are described in Section 4-A. Then, the experimental results of comparison with other multimodal algorithms are given in Section 4-B. Section 4-C analyzes the principle of CNMM from the experimental results. Finally, Section 4-D gives parameter analysis.

A.Benchmark Functions and Evaluation Protocols

To demonstrate the effectiveness of CNMM, we conducted experiments on

Table 1: PARAMETER SETTINGS

Test Function	$MaxFEs$	N
F_1-F_5	5.0E+04	80
F_6	2.0E+05	100
F_7	2.0E+05	300
F_8-F_9	4.0E+05	300
F_{10}	2.0E+05	100
F_{11}-F_{13}	2.0E+05	200
F_{14}-F_{20}	4.0E+05	200

Table 2: TEST FUNCTIONS

F1(Five-Uneven-Peak Trap)	F6(Shubert with 2D)	F11(Composition Function 1 with 2D)	F16(Composition Function 3 with 5D)
F2(Equal maxima)	F7(Vincent with 2D)	F12(Composition Function 1 with 2D)	F17(Composition Function 4 with 5D)
F3(Uneven Decreasing Maxima)	F8(Shubert with 3D)	F13(Composition Function 1 with 2D)	F18(Composition Function 3 with 10D)
F4(Himmelblau)	F9(Vincent with 3D)	F14(Composition Function 1 with 3D)	F19(Composition Function 4 with 10D)
F5(Six-Hump Camel Back)	F10(Modified Rastrigin)	F15(Composition Function 1 with 3D)	F20(Composition Function 1 with 20D)

a widely used benchmark function set—the CEC 2013 multimodal function set [24] containing 20 functions, which were designed for the 2013 IEEE CEC Special Session on Niching Methods for Multimodal Optimization.

The peak ratio(PR) and success rate(SR) were used as references [24] to evaluate the performance of the CNMM algorithm and compare it with other algorithms. The PR is the average number of optimal solutions found over all the runs divided by the known number of optimal solutions. A run is successful if all the optimal solutions have been found. The SR is the number of successful runs divided by the number of all the runs.

There were five kinds of precision in our experiments. They are $\varepsilon = 1.0E{-}01$, $\varepsilon = 1.0E-02$, $\varepsilon = 1.0E-03$, $\varepsilon = 1.0E-04$, and $\varepsilon = 1.0E-05$. But for $\varepsilon = 1.0E-01$ and $\varepsilon = 1.0E-02$, the algorithms achieved good results, so we use the accuracy $\varepsilon = 1.0E-04$ for comparison, which is also the accuracy used in references [25], [26], [27],[28], and [29]. To further illustrate the superiority of the CNMM algorithm, we also compared higher accuracy data ($\varepsilon = 1.0E-05$) with other algorithms. In order to ensure the fairness of the experiment, for

Table 3: THE IMPACT OF CNMM's MAIN STRATEGY ON RESULTS

Func	CNMM		CNMM-DE		CNMM-ES	
	PR	SR	PR	SR	PR	SR
F1	1.000	1.000	1.000 (≈)	1.000	1.000 (≈)	1.000
F2	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
F4	1.000	1.000	1.000 (≈)	1.000	1.000 (≈)	1.000
F5	1.000	1.000	1.000 (≈)	1.000	1.000 (≈)	1.000
F6	0.722	0.000	0.111 (+)	0.000	0.500 (+)	0.000
F7	0.000	0.000	0.000 (≈)	0.000	0.500 (-)	0.000
F8	0.209	0.000	0.000 (+)	0.000	0.110 (+)	0.000
F9	0.000	0.000	0.000 (≈)	0.000	0.116 (-)	0.000
F10	1.000	1.000	0.833 (+)	0.227	0.833 (+)	0.227
F11	1.000	1.000	0.667 (+)	0.000	1.000 (≈)	1.000
F12	0.750	0.078	0.625 (+)	0.000	0.725 (+)	0.000
F13	1.000	1.000	0.833 (+)	0.000	0.667 (+)	0.000
F14	0.667	0.000	0.167 (+)	0.000	0.667 (≈)	0.000
F15	0.500	0.000	0.000 (+)	0.000	0.500 (≈)	0.000
F16	0.667	0.000	0.000 (+)	0.000	0.657 (≈)	0.000
F17	0.125	0.000	0.000 (+)	0.000	0.125 (≈)	0.000
F18	0.000	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000
F19	0.000	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000
F20	0.000	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000
Significantly better (+)			10		5	
Significantly worse (-)			0		2	
Similar (≈)			10		13	

all the algorithms involved in the comparison, we set the common parameters of all algorithms to be the same. Table 1 lists the maximum number of fitness evaluations (MaxFEs) and population size (N) of each test function. The first ten test functions denoted as F_1-F_{10} in Table 2 are the commonly used test functions in the community of evolutionary multimodal optimization. The remaining ten test functions denoted as F_{11}-F_{20} in Table 2 are the composition functions. Note that all the test functions should be maximized. The details of these 20 test functions can be found in [24]. Furthermore, all experiments were carried out for 51 independent runs for statistics.

The algorithms were implemented in MATLAB 2016a, and executed using a computer with 4 Inter Core i5-6500 3.20 GHz CPUs and 16 GB memory. The operating system was Microsoft Windows 7. The amplification factor F and crossover rate CR in CNMM were 0.5 and 0.9, respectively. The parameter L was 3 in the close neighbor mobility strategy. Parameters e and r were 0.5 and 0.1 in the elite selection mechanism, respectively.

Table 4: EXPERIMENTAL RESULTS IN PR AND SR ON PROBLEMS F_1-F_{20} AT ACCURACY LEVEL $\varepsilon = 1.0E-04$

Func	CNMM		CDE		SDE		R2PSO		R3PSO		NSDE	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F1	1.000	1.000	1.000 (≈)	1.000	0.657 (+)	0.373	1.000 (≈)	1.000	1.000 (≈)	1.000	1.000 (≈)	1.000
F2	1.000	1.000	1.000 (≈)	1.000	0.737 (+)	0.529	1.000 (≈)	1.000	1.000 (≈)	1.000	0.776 (+)	0.667
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
F4	1.000	1.000	1.000 (≈)	1.000	0.284 (+)	0.000	0.946 (+)	0.784	0.966 (+)	0.863	0.240 (+)	0.000
F5	1.000	1.000	1.000 (≈)	1.000	0.922 (+)	0.843	1.000 (≈)	1.000	1.000 (≈)	1.000	0.745 (+)	0.490
F6	0.722	0.000	1.000 (-)	1.000	0.056 (+)	0.000	0.537 (+)	0.000	0.688 (-)	0.000	0.056 (+)	0.000
F7	0.000	0.000	0.861 (-)	0.000	0.054 (-)	0.000	0.484 (-)	0.000	0.436 (-)	0.000	0.053 (-)	0.000
F8	0.209	0.000	0.000 (+)	0.000	0.015 (+)	0.000	0.023 (+)	0.000	0.421 (-)	0.000	0.013 (+)	0.000
F9	0.000	0.000	0.474 (-)	0.000	0.011 (≈)	0.000	0.122 (-)	0.000	0.125 (-)	0.000	0.006 (≈)	0.000
F10	1.000	1.000	1.000 (≈)	1.000	0.147 (+)	0.000	0.905 (+)	0.353	0.850 (+)	0.000	0.098 (+)	0.000
F11	1.000	1.000	0.330 (+)	0.000	0.314 (+)	0.000	0.641 (+)	0.000	0.650 (+)	0.157	0.248 (+)	0.000
F12	0.750	0.078	0.002 (+)	0.000	0.208 (+)	0.000	0.932 (-)	0.000	0.537 (+)	0.000	0.135 (+)	0.000
F13	1.000	1.000	0.141 (+)	0.000	0.297 (+)	0.000	0.627 (+)	0.000	0.647 (+)	0.000	0.225 (+)	0.000
F14	0.667	0.000	0.026 (+)	0.000	0.216 (+)	0.000	0.408 (+)	0.000	0.637 (+)	0.000	0.190 (+)	0.000
F15	0.500	0.000	0.005 (+)	0.000	0.108 (+)	0.000	0.167 (+)	0.000	0.213 (+)	0.000	0.125 (+)	0.000
F16	0.667	0.000	0.000 (+)	0.000	0.000 (+)	0.000	0.095 (+)	0.000	0.431 (+)	0.000	0.170 (+)	0.000
F17	0.125	0.000	0.000 (+)	0.000	0.000 (+)	0.000	0.015 (≈)	0.000	0.096 (+)	0.000	0.108 (≈)	0.000
F18	0.000	0.000	0.167 (-)	0.000	0.167 (-)	0.000	0.036 (-)	0.000	0.101 (-)	0.000	0.163 (-)	0.000
F19	0.000	0.000	0.000 (≈)	0.000	0.105 (-)	0.000	0.000 (≈)	0.000	0.032 (-)	0.000	0.098 (-)	0.000
F20	0.000	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.002 (≈)	0.000	0.078 (-)	0.000	0.123 (-)	0.000
Significantly better(+)			8		14		9		9		12	
Significantly worse(-)			4		3		3		7		4	
Similar(≈)			8		3		8		4		4	

B.Compared With Multimodal Algorithms

The results of CNMM on F_1-F_{20} with respect to PR and SR at all accuracy levels (i.e,ε=1.E-01, ε=1.E-02, ε=1.E-03, ε=1.E-04, and ε=1.E-05) are given in Table 7.

To further evaluate the performance of CNMM, we compare the results obtained by CNMM with those obtained by several multimodal algorithms: CDE [30], SDE [23], R2PSO, R3PSO[22], NSDE[31], Self-CSDE [32], LOICDE,

Table 5: EXPERIMENTAL RESULTS IN PR AND SR ON PROBLEMS F_1-F_{20} AT ACCURACY LEVEL $\varepsilon = 1.0E-05$

Func	CNMM		CDE		SDE		R2PSO		R3PSO		NSDE	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F1	1.000	1.000	1.000 (≈)	1.000	0.657 (+)	0.373	1.000 (≈)	1.000	1.000 (≈)	1.000	1.000 (≈)	1.000
F2	1.000	1.000	1.000 (≈)	1.000	0.584 (+)	0.275	1.000 (≈)	1.000	1.000 (≈)	1.000	0.753 (+)	0.627
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
F4	1.000	1.000	0.755 (+)	0.431	0.284 (+)	0.000	0.907 (+)	0.627	0.966 (+)	0.863	0.235 (+)	0.000
F5	1.000	1.000	1.000 (≈)	1.000	0.853 (+)	0.706	1.000 (≈)	1.000	1.000 (≈)	1.000	0.608 (+)	0.235
F6	0.722	0.000	0.997 (-)	0.961	0.056 (+)	0.000	0.461 (+)	0.000	0.678 (+)	0.000	0.053 (+)	0.000
F7	0.000	0.000	0.699 (-)	0.000	0.054 (-)	0.000	0.427 (-)	0.000	0.405 (-)	0.000	0.053 (-)	0.000
F8	0.185	0.000	0.000 (+)	0.000	0.015 (+)	0.000	0.011 (+)	0.000	0.418 (-)	0.000	0.013 (+)	0.000
F9	0.000	0.000	0.397 (-)	0.000	0.011 (≈)	0.000	0.085 (-)	0.000	0.117 (-)	0.000	0.006 (≈)	0.000
F10	1.000	1.000	1.000 (≈)	1.000	0.147 (+)	0.000	0.843 (+)	0.118	0.832 (+)	0.118	0.098 (+)	0.000
F11	1.000	1.000	0.085 (+)	0.000	0.314 (+)	0.000	0.627 (+)	0.000	0.650 (+)	0.000	0.248 (+)	0.000
F12	0.750	0.078	0.000 (+)	0.000	0.208 (+)	0.000	0.353 (+)	0.000	0.529 (+)	0.000	0.135 (+)	0.000
F13	1.000	1.000	0.020 (+)	0.000	0.297 (+)	0.000	0.611(+)	0.000	0.647 (+)	0.000	0.225 (+)	0.000
F14	0.667	0.000	0.007 (+)	0.000	0.216 (+)	0.000	0.369 (+)	0.000	0.637 (+)	0.000	0.190 (+)	0.000
F15	0.500	0.000	0.000 (+)	0.000	0.108 (+)	0.000	0.150 (+)	0.000	0.208 (+)	0.000	0.125 (+)	0.000
F16	0.667	0.000	0.000 (+)	0.000	0.108 (+)	0.000	0.082 (+)	0.000	0.425 (+)	0.000	0.170 (+)	0.000
F17	0.125	0.000	0.000 (+)	0.000	0.076 (+)	0.000	0.010 (+)	0.000	0.096 (+)	0.000	0.108 (≈)	0.000
F18	0.000	0.000	0.167 (-)	0.000	0.026 (-)	0.000	0.033 (-)	0.000	0.101 (-)	0.000	0.163 (-)	0.000
F19	0.000	0.000	0.000 (≈)	0.000	0.105 (-)	0.000	0.000 (≈)	0.000	0.032 (-)	0.000	0.098 (-)	0.000
F20	0.000	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.074 (-)	0.000	0.123 (-)	0.000
Significantly better			9		14		11		10		12	
Significantly worse			4		3		3		6		4	
Similar			7		3		6		4		4	

LOISDE[33] and LIPS [15]. The results of these multimodal algorithms come from the supplementary materials of [34], which were obtained under the same MaxFEs.

Tables 4, 5, and 6 show the results of CNMM and the other algorithms with PR and SR values accuracy level of ε=1.E-04 and ε=1.E-05, respectively. In addition, Wilcoxon's rank sum test [35] at $\alpha = 0.05$ with respect to PR between CNMM and other multimodal algorithms was performed to evaluate the statistical significance of the results. The symbols "+", "-"and "≈" indicate CNMM performed significantly better (+), significantly worse (-) or similarly (≈). We analyze the data from the experiments as follows.

Table 6: EXPERIMENTAL RESULTS IN PR AND SR ON PROBLEMS F_1-F_{20} AT ACCURACY LEVEL $\varepsilon = 1.0E-04$

Func	CNMM		Self-CSDE		LOICDE		LOISDE		LIPS	
	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	0.833 (+)	0.686
F2	1.000	1.000	1.000 (≈)	1.000	1.000 (≈)	1.000	0.235 (+)	0.039	1.000 (≈)	1.000
F3	1.000	1.000	1.000 (≈)	1.000	1.000 (≈)	1.000	1.000 (≈)	1.000	0.961 (+)	0.961
F4	1.000	1.000	0.686 (+)	0.294	0.975 (+)	0.902	0.250 (+)	0.000	0.990 (≈)	0.961
F5	1.000	1.000	0.961 (+)	0.922	1.000 (≈)	1.000	0.667 (+)	0.333	1.000 (≈)	1.000
F6	0.722	0.000	0.699 (+)	0.020	1.000 (-)	1.000	0.056 (+)	0.000	0.246 (+)	0.000
F7	0.000	0.000	0.695 (-)	0.000	0.705 (-)	0.020	0.029 (-)	0.000	0.400 (-)	0.000
F8	0.209	0.000	0.695 (-)	0.000	0.000 (+)	0.000	0.012 (+)	0.000	0.084 (+)	0.000
F9	0.000	0.000	0.265 (-)	0.000	0.187 (-)	0.000	0.005 (-)	0.000	0.104 (-)	0.000
F10	1.000	1.000	0.992 (+)	0.992	1.000 (≈)	1.000	0.083 (+)	0.000	0.748 (+)	0.000
F11	1.000	1.000	0.339 (+)	0.000	0.660 (+)	0.000	0.167 (+)	0.000	0.974 (+)	0.843
F12	0.750	0.078	0.321 (+)	0.000	0.495 (+)	0.000	0.125 (+)	0.000	0.574 (+)	0.000
F13	1.000	1.000	0.317 (+)	0.000	0.510 (+)	0.000	0.167 (+)	0.000	0.794 (+)	0.176
F14	0.667	0.000	0.304 (+)	0.000	0.657 (≈)	0.000	0.167 (+)	0.000	0.644 (+)	0.000
F15	0.500	0.000	0.186 (+)	0.000	0.299 (+)	0.000	0.125 (+)	0.000	0.336 (+)	0.000
F16	0.667	0.000	0.072 (+)	0.000	0.559 (+)	0.000	0.167 (+)	0.000	0.304 (+)	0.000
F17	0.125	0.000	0.056 (+)	0.000	0.223 (-)	0.000	0.076 (+)	0.000	0.162 (-)	0.000
F18	0.000	0.000	0.003 (≈)	0.000	0.219 (-)	0.000	0.157 (-)	0.000	0.098 (-)	0.000
F19	0.000	0.000	0.000 (≈)	0.000	0.037 (-)	0.000	0.027 (-)	0.000	0.000 (≈)	0.000
F20	0.000	0.000	0.000 (≈)	0.000	0.123 (-)	0.000	0.088 (-)	0.000	0.000 (≈)	0.000
Significantly better(+)			11		8		13		12	
Significantly worse (-)			4		6		5		4	
Similar(≈)			5		6		2		4	

- 1) **Test functions $\mathbf{F_1}$-$\mathbf{F_5}$.** Due to the small number of decision variables and the low number of global optimal solutions, the most important thing is that the slope of each mountain is relatively flat. CNMM is able to find all global optimal solutions stably every time. From tables 4, 5, and 6, it can be seen the CNMM performs significantly better on F_1-F_5 than most of the algorithms, no matter at which accuracy level(See Table 7). In addition, CDE, NCDE, PNPCDE, and Self-CCDE all performed well in the first five test functions. This is because the DE algorithm is extended with a crowding scheme making it capable of tracking and maintaining multiple optima.

- 2) **Test functions $\mathbf{F_6}$-$\mathbf{F_9}$ and $\mathbf{F_{17}}$-$\mathbf{F_{20}}$.** For test functions F_6-F_9 with many spikes, CNMM performed worse than most other algorithms. This is because, for many sharp peaks, the range of decision space occupied by the spikes is small, and it is difficult for our particles to find the position where the peaks are located, so there is no possibility that the particles lead the surrounding particles to the peaks. For test problems F_{17} to F_{20}, the number of decision variables reaches 10 and 20. For PSO, when the number of decision variables reaches a certain number, all decision variables affect the final function value. Therefore, it is difficult to find the true particles close to the mountain based on the merits of the function values. In other words, it is difficult to find the real pbest and gbest, so it is difficult for the particles to reach the true mountain.

- 3) **Test functions $\mathbf{F_{10}}$-$\mathbf{F_{16}}$.** On the test functions F_{10}-F_{16}, which have many relatively flat peaks, the CNMM had good results. On F_{10}, F_{11} and F_{13}, CNMM was better than most algorithms. This is because in the CNMM algorithm, if the particles are distributed over some gentle peaks, these excellent particles cause the surrounding particles to reach the highest peak, and due to the existence of the DE strategy, the hard-to-find peaks can also be somewhat explored by particles in the population. Therefore, CNMM showed extraordinary vitality in these test functions.

- 4) **Data comparison for different levels of accuracy.** Comparing Table 4 with Table 5, we can clearly see that when the accuracy is improved to ε=1.E-05, the difference between the CNMM algorithm and CDE, R2PSO, and R3PSO is further increased. Therefore, the data after the CNMM algorithm is run is very close to the true peak. This is because the particles are on the same mountain, and the particles are constantly exploring the top of the mountain, constantly fine-tuning the operation, and every generation of elite particles are preserved so that the population does not degenerate. Therefore, the higher the accuracy of the CNMM algorithm, the more prominent the effect.

Table 7 shows the raw data of the CNMM algorithm on all accuracy levels. As can be seen from Table 7, for the test functions F_6, F_{12}, F_{14}, F_{15}, F_{16} and F_{17}, the PR values are the same on the accuracy levels $\varepsilon = 1.0E-01$, $\varepsilon = 1.0E-02$, $\varepsilon = 1.0E-03$,$\varepsilon = 1.0E-04$, and $\varepsilon = 1.0E-05$, which means that the last solution obtained by the CNMM has very high precision. In other words, as long as a particle has explored a mountain, the highest height of the mountain will be found by the particles. It also illustrates the advantages of CNMM over the other algorithms in terms of accuracy.

Overall, we can conclude the CNMM algorithm generally outperformed most of the multimodal algorithms in terms of PR and SR. Moreover, the CNMM algorithm had a very obvious advantage in approaching the highest peak. In addition,the DE strategy is used in CNMM. This helps the algorithms to keep population diversity in order to locate more global optima and accelerate the convergence speed to improve the accuracy of solutions. Therefore, the CNMM algorithm outperformed the other multimodal algorithms.

C.Effects of CNMM Components

The main components of the CNMM algorithm are 1) Elite Selection Mechanism; 2) Close Neighbor Mobility Strategy; and 3) Modified DE Strategy. Here we discuss the impact and principles of each strategy. In addition, the impact of

Table 7: EXPERIMENTAL RESULTS IN PR AND SR OF CNMM ON 20 PROBLEMS F_1-F_{20} AT ALL FIVE ACCURACY LEVELS

CNMM										
ε	F1		F2		F3		F4		F5	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
1E-01	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1E-02	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1E-03	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1E-04	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1E-05	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
e	F6		F7		F8		F9		F10	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
1E-01	0.722	0.000	0.750	0.000	0.246	0.000	0.050	0.000	1.000	1.000
1E-02	0.722	0.000	0.028	0.000	0.222	0.000	0.000	0.000	1.000	1.000
1E-03	0.722	0.000	0.000	0.000	0.209	0.000	0.000	0.000	1.000	1.000
1E-04	0.722	0.000	0.000	0.000	0.209	0.000	0.000	0.000	1.000	1.000
1E-05	0.722	0.000	0.000	0.000	0.185	0.000	0.000	0.000	1.000	1.000
ε	F11		F12		F13		F14		F15	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
1E-01	1.000	1.000	0.750	0.078	1.000	1.000	0.667	0.000	0.500	0.000
1E-02	1.000	1.000	0.750	0.078	1.000	1.000	0.667	0.000	0.500	0.000
1E-03	1.000	1.000	0.750	0.078	1.000	1.000	0.667	0.000	0.500	0.000
1E-04	1.000	1.000	0.750	0.078	1.000	1.000	0.667	0.000	0.500	0.000
1E-05	1.000	1.000	0.750	0.078	1.000	1.000	0.667	0.000	0.500	0.000
ε	F16		F17		F18		F19		F20	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
1E-01	0.667	0.000	0.125	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1E-02	0.667	0.000	0.125	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1E-03	0.667	0.000	0.125	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1E-04	0.667	0.000	0.125	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1E-05	0.667	0.000	0.125	0.000	0.000	0.000	0.000	0.000	0.000	0.000

the number of neighbors on the experimental results is analyzed in this section.

1)*Elite Selection Mechanism:* In Table 3, CNMM-DE represents the result of the CNMM algorithm with the modified DE strategy, and CNMM-ES represents the result of the CNMM algorithm to remove the elite selection mechanism. The excellent PR value uses a black background, the symbols "+", "-"and "≈" indicate the CNMM algorithm performed significantly better (+), significantly worse (-) or similarly (≈). We can clearly see from the table that after removing the elite retention mechanism, there are nine test functions that are worse than before, but there are also two test functions that performed better than the full algorithm. This is because the test functions F_7 and F_9 have multiple sharp peaks, and the elite retention mechanism does not update the elite particles, thus reducing the population's exploration area. However on most of the test functions, the elite retention mechanism has shown good results. This is because the elite retention mechanism ensures that the population does not degenerate and constantly directs particles around itself to approach the current peak. The inspiration for the elite retention strategy comes from X. Li [24]. In the course of the experiment, r takes 0.1 and e takes 0.5 (see Algorithm 1).

2)*Close Neighbor Mobility Strategy:*The Close Neighbor Mobility Strategy is the core strategy of CNMM. Influenced by the idea of ring topology, this paper designs a unique niche method. Each particle and the nearest three particles form a special niche. Each particle moves to the niche in its own niche with the highest value of the objective function. This allows each particle to have the highest efficiency of movement. Since each particle constitutes a niche only in the last three particles, each population will have multiple niches, which is beneficial to find the most global optimal solutions. A particle with a high target function value will become the leader of the surrounding particle. In this way, the particles move in the direction of the mountain, which is conducive to constantly approaching the highest peak.

3)*Modified DE Strategy:*From Table 3, we can clearly see that the effect is obviously worse when the CNMM algorithm removes the DE strategy. The results with 10 test functions got worse because when the program evolved into

Table 8: IN ALGORITM 2 OF CNMM, THE INFLUENCE OF PARAMETER L ON THE RESULT IS IN THE TEST FUNCTION $F_1 - F_{20}$ AT ACCURACY LEVEL $\varepsilon = 1.0E-04$

Func	L=3		L=4		L=5		L=6		L=7		L=8		L=9	
	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
F2	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
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
F4	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
F5	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
F6	0.722	0.000	0.500 (+)	0.000	0.611 (+)	0.000	0.500 (+)	0.000	0.330 (+)	0.000	0.389 (+)	0.000	0.500 (+)	0.000
F7	0.000	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.027 (-)	0.000
F8	0.209	0.000	0.160 (+)	0.000	0.185 (+)	0.000	0.168 (+)	0.000	0.148 (+)	0.000	0.123 (+)	0.000	0.197 (≈)	0.000
F9	0.000	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000
F10	1.000	1.000	1.000 (≈)	1.000	1.000 (≈)	1.000	1.000 (≈)	1.000	0.917 (+)	0.000	0.750 (+)	0.000	1.000 (≈)	1.000
F11	1.000	1.000	1.000 (≈)	1.000	1.000 (≈)	1.000	1.000 (≈)	1.000	0.750 (+)	0.000	1.000 (≈)	1.000	1.000 (≈)	1.000
F12	0.750	0.078	0.750 (≈)	0.000	0.875 (-)	0.125	0.625 (+)	0.000	0.667 (+)	0.000	0.750 (≈)	0.000	0.750 (≈)	0.000
F13	1.000	0.000	0.833 (+)	0.000	0.833 (+)	0.000	0.833 (+)	0.000	0.667 (+)	0.000	0.667 (+)	0.000	0.667 (+)	0.000
F14	0.667	0.000	0.667 (≈)	0.000	0.667 (≈)	0.000	0.667 (≈)	0.000	0.500(+)	0.000	0.667 (≈)	0.000	0.667 (≈)	0.000
F15	0.500	0.000	0.500 (≈)	0.000	0.500 (≈)	0.000	0.375 (+)	0.000	0.667 (-)	0.000	0.500 (≈)	0.000	0.375 (+)	0.000
F16	0.667	0.000	0.667 (≈)	0.000	0.667 (≈)	0.000	0.667 (≈)	0.000	0.250 (+)	0.000	0.500 (+)	0.000	0.500 (+)	0.000
F17	0.125	0.000	0.125 (≈)	0.000	0.125 (≈)	0.000	0.250 (-)	0.000	0.000 (+)	0.000	0.125 (≈)	0.000	0.250 (-)	0.000
F18	0.000	0.000	0.000 (≈)	0.000	0.167 (-)	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.167 (-)	0.000	0.167 (-)	0.000
F19	0.000	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000
F20	0.000	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000	0.000 (≈)	0.000
Significantly better(+)			3		3		5		9		5		4	
Significantly worse (-)			0		2		1		1		1		3	
Similar (≈)			17		15		14		10		14		13	

a certain algebra, all the particles gathered on their nearest peaks. Due to the aggregation effect of the algorithm, there were some particles on the mountain that did not distribute the particles, which led to the algorithm falling into local optimum. In order to solve this problem, when the program runs to a certain algebra, we use the DE strategy to update the particles with a low function value in the whole population. This way the population can explore as many areas as possible. The data in Table 3 demonstrate the impact of the DE strategy.

D.Parameter analysis

1) *The effect of the number of neighbors on the results:* Parameter L in Algorithm 2 is important because it indicates the number of neighbors per particle. Generally speaking, it is important to understand the influence of control parameters on the performance of a novel evolutionary algorithm. Here, we investigate how L affects the effectiveness of the CNMM algorithm.

In Table 8, the data in the shaded part show the best data in the CNMM algorithm results when the L parameters are different. The symbols "+", "-"and "≈" indicate the CNMM algorithm($L = 3$) performed significantly better (+), significantly worse (-) or similarly (≈). The last line shows the number of best results among the 20 test functions. We can clearly see that as the value of the L parameter continues to increase, the effect of the CNMM is also constantly changing. But it is not difficult to see that when parameter L is 3, the CNMM algorithm had the best results. This is because for multimodal problems, the algorithm should be able to find more global optimal solutions. When the number of neighbors per particle is small, the particle-guided range with excellent fitness is smaller, so different particles are better able to find different global optimal solutions.

Because L represents the number of neighbors in the small group formed by each particle, when L is greater than 10, the number of people in each small group reaches a certain scale. Once an excellent particle appears, the rest of the other particles containing this particle will move to the position of this excellent particle. In this way, the influence of a single excellent particle is so large that

the whole population can not find all the global optimal solutions. From the experimental results, when L is greater than 10, the effect of CNMM algorithm is very bad, so only when L is less than 10 is it listed in the figure.

2)*Analysis of mutation frequency*

In Algorithm 4, parameter K controls the frequency of population updates. The value of K ranges from 0 to 1. During the whole process of particle swarm evolution, the larger the K, the fewer the number of particle group updates, and the smaller the K, the more the particle group is updated. The way to update the population is to use the modified DE strategy for particles that are not in the S set. The purpose of updating the population is to jump out of the local optimum and find more global optimal solutions. If the frequency of the entire population update is too high, it is difficult to ensure that the particles are close to the highest peak. In order to balance the diversity and convergence of the whole population, the selection of K value must be reasonable. In the CNMM algorithm, the K value of 0.2 can obtain the best results. The K value is an empirical value.

3)*CNMM algorithm in test function* F_{10}

Fig.7 shows the details of the operation of the CNMM algorithm on F_{10}. In Fig.7.A, the initial population is randomly generated in the decision space. We can find the random distribution of particles in the decision space. After 15 generations, as shown in Fig.7.B, the particles begin to clump, and the particles move toward the relatively close and excellent particle direction. After 30 generations, as shown in Fig.7.C, the particles have basically moved to their nearest peaks. After 45 generations, as shown in Fig.7.D, the particles almost all clustered on different peaks. After about 100 generations, as shown in Fig.7.H, we can see that almost all the particles overlap at the highest point of their respective peaks. From the continuous progression, we can easily see that the CNMM algorithm has high accuracy.

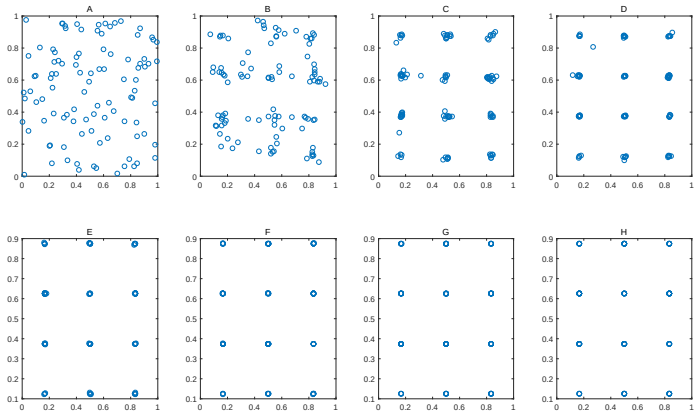


Figure 7: CNMM algorithm in the F_{10} test function running process

5. Conclusion

This paper has developed a particle swarm optimizer using the close neighbor mobility strategy for solving MMOPs, which can achieve a better balance between exploration and exploitation. Three novel techniques have been developed to improve the performance of the algorithm: 1)elite selection mechanism; 2)close neighbor mobility strategy; and 3)modified DE strategy.

The elite selection strategy ensures that the population does not degenerate and leads the surrounding particles to search. Furthermore, the particles of each special small group of the neighboring mobility strategy rush to the highest peak, and quickly reach the top of the mountain, effectively ensuring the accuracy of the algorithm. Finally, the modified DE strategy constantly explores new locations in space and tries to explore more peaks to ensure the distribution of the population.

Based on these three novel techniques, CNMM can achieve a promising performance when dealing with MMOPs, regardless of the accuracy level. The results also show the efficifency and feasibility of the CNMM for quickly locating more accurate global optima.

The future work includes five aspects.

- 1) In the elite selection strategy of the CNMM algorithm, we introduced two parameters e and r, whose purpose is to find out all the particles which are closest to the highest peak and have good distribution in the current population. In this paper, e and r adopt fixed values. The future work is to adaptively adapt the parameters according to the actual situation of evolution.

- 2) In the DE strategy, in order to make the combination of DE strategy and the other two strategies achieve better results, the CNMM algorithm makes a small modification of the DE strategy. However, this modification did not achieve surprising results, so we can consider designing a more powerful search engine based on DE in the future.

- 3) In close neighbor mobility strategy, we can design some more interesting methods to measure the position relationship between particles.

- 4) We are considering the application of CNMM to a few real-world MMOPs, such as electromagnetic optimization, game optimization, production scheduling, and resource allocation.

- 5) Because the CNMM algorithm has very high accuracy, we can combine CNMM with other algorithms, and maybe get better comprehensive performance.

6. Acknowledgements

The authors wish to thank the support of the National Natural Science Foundation of China(Grant No. 61876164, 61673331, 61772178), the Education Department Major Project of Hunan Province(Grant No 17A212), the Science and Technology Plan Project of Hunan Province(Grant No. 2018TP1036,2016TP1020), the Provinces and Cities Joint Foundation Project(Grant No. 2017J04001).

References

[1] A. Y. Goharrizi, R. Singh, A. M. Gole, S. Filizadeh, J. C. Muller, R. P. Jayasinghe, A parallel multimodal optimization algorithm for simulation-based design of power systems, Vol. 30, IEEE, 2015, pp. 2128–2137.

[2] K.-C. Wong, K.-S. Leung, M.-H. Wong, Protein structure prediction on a lattice model via multimodal optimization techniques, in: Proceedings of the 12th annual conference on Genetic and evolutionary computation, ACM, 2010, pp. 155–162.

[3] W. Sheng, S. Swift, L. Zhang, X. Liu, A weighted sum validity function for clustering with a hybrid niching genetic algorithm, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 35 (6) (2005) 1156–1167.

[4] M. Boughanem, L. Tamine, A study on using genetic niching for query optimisation in document retrieval (2002) 135–149.

[5] I. BoussaïD, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Information sciences 237 (2013) 82–117.

[6] S. W. Mahfoud, Niching methods for genetic algorithms, Ph.D. thesis, Citeseer (1995).

[7] K. D. Koper, M. E. Wysession, D. A. Wiens, Multimodal function optimization with a niching genetic algorithm: A seismological example, Bulletin of the Seismological Society of America 89 (4) (1999) 978–988.

[8] J. rey Horn, N. Nafpliotis, D. E. Goldberg, A niched pareto genetic algorithm for multiobjective optimization, in: Proceedings of the first IEEE conference on evolutionary computation, IEEE world congress on computational intelligence, Vol. 1, Citeseer, 1994, pp. 82–87.

[9] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE transactions on evolutionary computation 8 (3) (2004) 204–210.

[10] D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, M. N. Vrahatis, Parallel differential evolution, in: Proceedings of the 2004 congress on evolutionary computation (IEEE Cat. No. 04TH8753), Vol. 2, IEEE, 2004, pp. 2023–2029.

[11] M. Weber, F. Neri, V. Tirronen, Distributed differential evolution with explorative–exploitative population families, Genetic Programming and Evolvable Machines 10 (4) (2009) 343.

[12] M. Weber, V. Tirronen, F. Neri, Scale factor inheritance mechanism in distributed differential evolution, Soft Computing 14 (11) (2010) 1187–1207.

[13] S. Das, A. Konar, U. K. Chakraborty, A. Abraham, Differential evolution with a neighborhood based mutation operator: a comparative study, IEEE Transactions on Evolutionary Computation 13 (3).

[14] U. K. Chakraborty, S. Das, A. Konar, Differential evolution with local neighborhood, in: 2006 IEEE International Conference on Evolutionary Computation, IEEE, 2006, pp. 2042–2049.

[15] B.-Y. Qu, P. N. Suganthan, S. Das, A distance-based locally informed particle swarm model for multimodal optimization, IEEE Transactions on Evolutionary Computation 17 (3) (2012) 387–402.

[16] 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.

[17] S. Biswas, S. Kundu, S. Das, An improved parent-centric mutation with normalized neighborhoods for inducing niching behavior in differential evolution, IEEE transactions on cybernetics 44 (10) (2014) 1726–1737.

[18] S. Biswas, S. Kundu, S. Das, Inducing niching behavior in differential evolution through local information sharing, IEEE Transactions on Evolutionary Computation 19 (2) (2014) 246–263.

[19] B.-Y. Qu, P. N. Suganthan, S. Das, A distance-based locally informed particle swarm model for multimodal optimization, IEEE Transactions on Evolutionary Computation 17 (3) (2013) 387–402.

[20] X. Li, Niching without niching parameters: particle swarm optimization using a ring topology, IEEE Transactions on Evolutionary Computation 14 (1) (2010) 150–169.

[21] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, Journal of global optimization 11 (4) (1997) 341–359.

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

[23] X. Li, Efficient differential evolution using speciation for multimodal function optimization, in: Proceedings of the 7th annual conference on Genetic and evolutionary computation, ACM, 2005, pp. 873–880.

[24] X. Li, A. Engelbrecht, M. G. Epitropakis, Benchmark functions for cec'2013 special session and competition on niching methods for multimodal function optimization, RMIT University, Evolutionary Computation and Machine Learning Group, Australia, Tech. Rep.

[25] 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) (2017) 894–908.

[26] 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.

[27] S. Hui, P. N. Suganthan, Ensemble and arithmetic recombination-based speciation differential evolution for multimodal optimization, IEEE transactions on cybernetics 46 (1) (2015) 64–74.

[28] Y. Wang, H.-X. Li, G. G. Yen, W. Song, Mommop: Multiobjective optimization for locating multiple optimal solutions of multimodal optimization problems, IEEE transactions on cybernetics 45 (4) (2014) 830–843.

[29] Q. Yang, W.-N. Chen, Z. Yu, T. Gu, Y. Li, H. Zhang, J. Zhang, Adaptive multimodal continuous ant colony optimization, IEEE Transactions on Evolutionary Computation 21 (2) (2016) 191–205.

[30] R. Thomsen, Multimodal optimization using crowding-based differential evolution, in: Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753), Vol. 2, IEEE, 2004, pp. 1382–1389.

[31] 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.

[32] 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.

[33] 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.

[34] Q. Yang, W.-N. Chen, Y. Li, C. P. Chen, X.-M. Xu, J. Zhang, Multimodal estimation of distribution algorithms, IEEE transactions on cybernetics 47 (3) (2017) 636–650.

[35] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation 1 (1) (2011) 3–18.

[36] C. Yue, B. Qu, J. Liang, A multiobjective particle swarm optimizer using ring topology for solving multimodal multiobjective problems, IEEE Transactions on Evolutionary Computation 22 (5) (2017) 805–817.