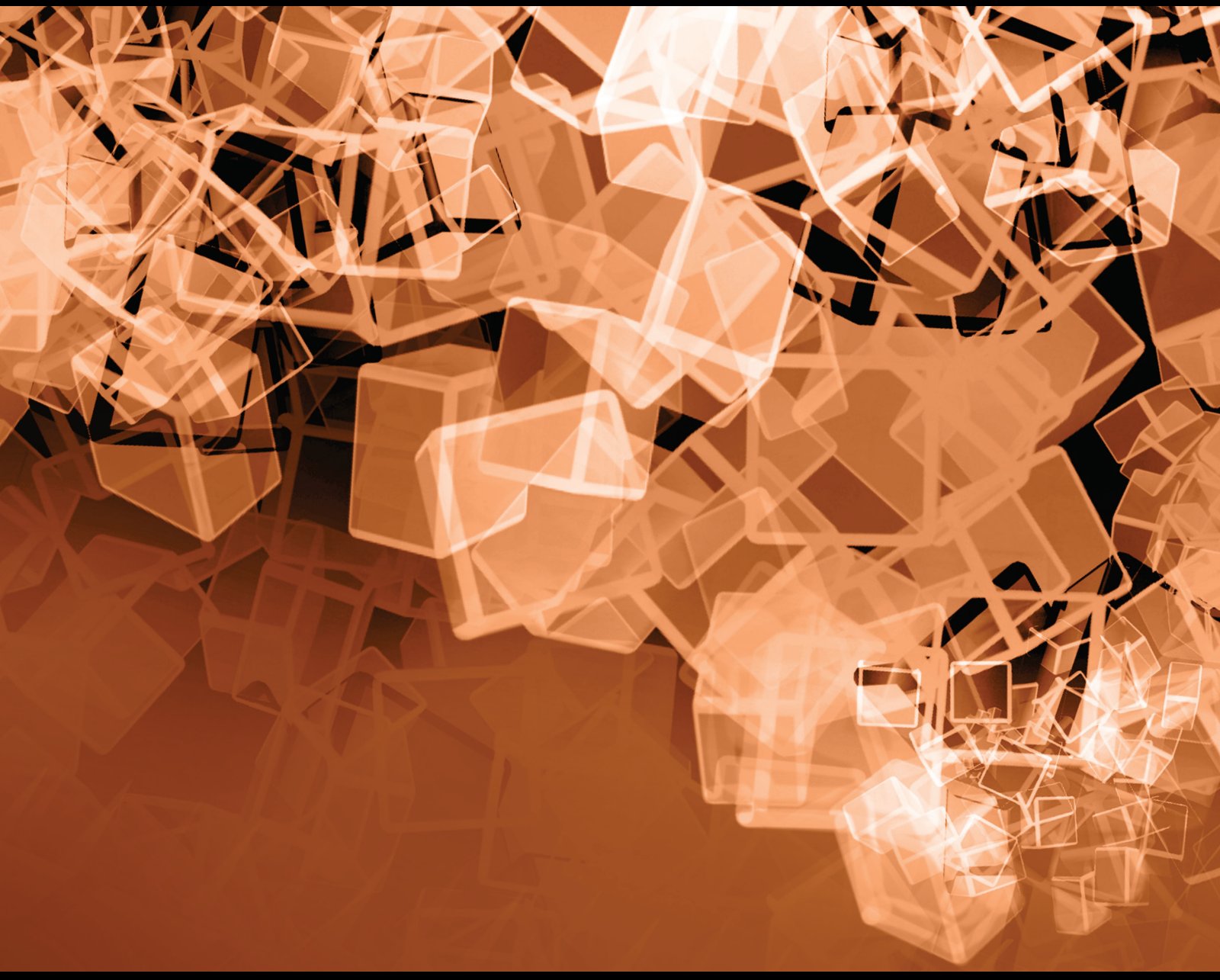
Discrete Dynamics in Nature and Society

Data Analysis and Modeling for Complex Swarm Intelligence Systems

Lead Guest Editor: Shi Cheng
Guest Editors: Guo Yu, Mohammed El-Abd, and Xueyi Wang



Data Analysis and Modeling for Complex Swarm Intelligence Systems

Discrete Dynamics in Nature and Society

Data Analysis and Modeling for Complex Swarm Intelligence Systems

Lead Guest Editor: Shi Cheng
Guest Editors: Guo Yu, Mohammed El-Abd, and Xueyi Wang

Contents

Hindawi
Discrete Dynamics in Nature and Society
Volume 2023, Article ID 8679509, 10 pages
https://doi.org/10.1155/2023/8679509

Research Article

A Degenerate Primer Design Method Based on Ant Colony Optimization

Xianghe Wang, Jiaxu Ning, Yueqiu Jiang, Ruhui Liu, Xinyuan Zhang, and Jiaxuan Wu

School of Information Science and Engineering, Shenyang Ligong University, Shenyang, Liaoning, China

Correspondence should be addressed to Jiaxu Ning; ningjiaxu@sylu.edu.cn and Yueqiu Jiang; jiangyueqiu_sylu@163.com

Received 27 June 2022; Revised 30 July 2022; Accepted 10 August 2022; Published 18 April 2023

Academic Editor: Shi Cheng

For the existed biological degenerate primer design problem, a solution method based on an ant colony optimization algorithm is proposed, in which an optimization model considers primer coverage as well as degeneracy while allowing for a small number of base mismatches. Using the construction graph model, the algorithm makes full use of the dynamic change information during the iterative optimal path optimization process, strengthens the explored new path pheromone, and selects the primers that meet the constraints. The results of DNA template sequence matching experiments show that the degenerate primers designed by the proposed algorithm outperform existing approaches.

1. Introduction

Degenerate primers are a mixture of different sequences encoding all different base probabilities of a single amino acid, and their degeneracy is defined as the number of combinations of all sequences at one or more positions in a PCR primer [1]. They are the primers with a length of k, whose degeneracy is most at d, matching at least m strings in a set of n strings (m represents the coverage of the primer).

Under the condition of certain specificity, degenerate primers should have the lowest degeneracy and the highest coverage. In PCR primer design, the length of primers is mostly fixed; therefore, only coverage degree m and degeneracy degree d need to be optimized [2–5]. However, as a consequence of incoherence between sequences and inadequacies in the fitness evaluation function, this method frequently results in high degeneracy of primers. Therefore, while designing primer pairs, some rather than all of the target DNA sequences are usually chosen to match. Consequently, this paper focuses on a degenerate primer design (DPD) problem that minimizes the degeneracy and maximizes coverage while allowing for a few mismatches. For a primer S, if its uncovered degree and degeneracy are computed by functions $\Gamma(S)$ and $\Psi(S)$, then its quality can be computed by the following formula.

$$F(S) = mu\Gamma(S) + \text{phi}\psi(S). \tag{1}$$

The mu and phi are weight parameters used to adjust the importance of coverage and degeneracy. The degeneracy degree is the total number of sequence combinations contained within the primer at one or more positions of a degenerate PCR. The computational procedure of the degeneracy degree is described in detail in [6]. Moreover, the constraints [11] related with primer length, primer pair differential length, primer melting temperature, primer pair differential melting temperature, GC content, $3'$end constraint, hairpin structure constraint, and double chain structure constraint are also considered in this work.

The existing related methods for this problem mostly are classical heuristic ways. For example, dynamic pattern matching [7], two-phase MIPS algorithms [8], iterative beam search [9], and HYDEN are based on the approximate algorithm [15]. These methods have been included in some well-known primer design tools [12–14], and some of them have been parallelized [16]. In recent years, swarm intelligence

optimization algorithms [17–21] have proven to be an effective way to tackle difficult optimization problems. The genetic algorithm is also used for the primer design problem, such as the single PCR design [10] and multiple PCR design [11], and get competitive performance. But few swarm intelligence algorithms have been applied to solve the constraint DPD problem.

This research presents an original way to DPD using an ant colony optimization algorithm with an enhanced pheromone update mechanism. With the help of the construction graph model, the constraints of DPD are dynamically satisfied during the construction process. The primer pairs are finally screened by analyzing and comparing the two indexes of degeneracy and coverage with existing methods.

2. Ant Colony Optimization Model for Degenerate Primer Design Problem

In this paper, an enhanced pheromone updating mechanism is proposed for ant colony optimization—ACOH. To optimize the pheromone update and improve convergence speed without compromising quality, the algorithm fully utilizes the dynamic information used in the iterative optimal path optimization process.

To the original pheromone matrix, an incremental matrix is added. All ants in a colony share the same pheromone incremental matrix. IOSolution represents the optimal (best) solution set so far; BIOSolution represents the better (better) solution set, and VInformation represents the differences between them. Using the pheromone dynamic update mechanism can improve the efficiency of the solution based on the existing pheromone update mechanism.

The algorithm generates an iterative optimal solution, known as the IOSolution, through several iterations and then compares it with the iterative better solution BIOSolution generated by the algorithm determining the dynamic change information VInformation, which indicates which path side arcs are newly explored. Then, when the original pheromone is updated, we apply additional dynamic pheromone enhancement to this modified information. The increment is stored in the pheromone increment matrix and set to the reciprocal of the iterative better path fitness evaluation function value. In this way, the probability of ants visiting the side arc in the future may be improved, and the diversity of understanding can be expanded to some level; consequently, the run time of the algorithm is reduced, improving its convergence rate and solution efficiency.

The method, with the assistance of the construction graph model (CGM), used ACOH to complete the design of degenerate primers. The algorithm focuses on DPD with maximum coverage and minimum degeneracy.

The number of layers of the CGM depends on the length of primers and generally ranges from 18 to 26 layers. After the 18th layer, an empty node is set in each layer, which facilitates completing the primer design and exiting the program. Each layer is a combination of 15 base pairs including [ATGC], [GTA], [GCA], [GCT], [CTA], [GT], [GA], [CG], [CA], [CT], [AT], [A], [T], [G], and [C]. The final combination must meet a number of limitations and basic pairing requirements and iterate repeatedly to obtain the best primer pair, or it stops here at the maximum iteration after a series of iterations. With the restrictions of coverage and degeneracy, as well as the principles of base pairing, each layer selects the appropriate base combination branch. It iterates circularly until the suitable primer pair is searched, and the results are output, or the maximum iteration time is reached. Then, the output results are screened and filtered by the constraints of multiple primer designs. The primer pairs that finally meet the requirements of constraints and various parameters are output, such as position, length, annealing temperature of up primers and down primers, the content of GC in primer pairs, the judgment of complementarity and specificity between primers, and so on.

The algorithm consists of five stages: initialization, constructing solutions, updating pheromones, evaluating solutions, and judging the outcome of the algorithm. The whole process of solving is shown in Figure 1.

In order to make the ACOH algorithm applicable to DPD, the initialization, constructing solutions, and judging the outcome of the algorithm were redesigned accordingly, and the other parts were completed using the methods in the ACO algorithm [22]. The details of the redesign are as follows:

In the construction of the initial graph model, each ant randomly selects an initial node and has fifteen choices for the following node. According to the input template DNA sequence and base pairing principle, ants roughly screen out the required base combinations. Following that, a roulette wheel is used to select the next node that will be visited using the probability state transition formula. Here, the ant constructs the solution matrix using a three-dimensional matrix, because in the first 18 nodes, each node initially has 15 choices. To distinguish one node from the same base combination in another node, a sentinel identification bit is specially established. For example, the third node has a base combination of [GC], and the fourth node also has a base combination of [GC]. Some ants are just on the second node, facing the same base combination [GC], which makes it hard to distinguish. At this point, the sentry of the identification position plays a key role in the primer design process, and the third node is selected to ensure order. In this way, these nodes are selected in turn until the 18th node is reached. The length of primers is generally 18–26 bp; therefore, an empty node is set on the 18th through 26th base combinations in addition to the above 15 combinations, which facilitates the exit of the path construction at any time after one iteration. After that, each ant completes the path construction once, which represents the completion of the design of the initial primer pair. Then, each path is evaluated to screen out an iterative optimal path. The specific process will be introduced in detail in the later evaluation stage. After the pheromone is updated in this path, the next iteration is continued. Repeat until the maximum amount of iterations has been reached, or fitness requirements have been met.

In the process of degenerate primer design, the introduction to the second part reveals that constraints are crucial

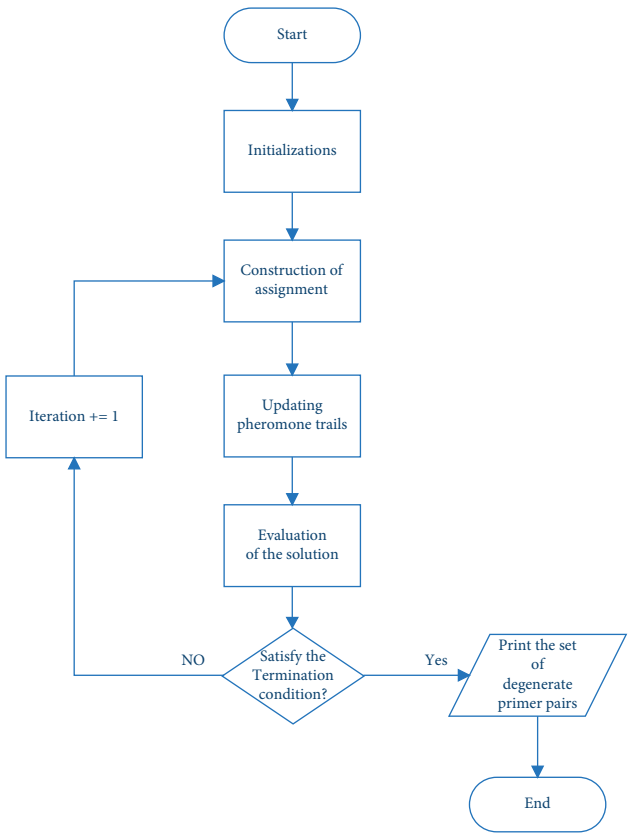


Figure 1: ACOH for solving DPD.

for the final primer pair design. Hard constraint conditions and soft constraint conditions are the two types of constraint conditions, with the hard constraint condition referring to the need that the primer pair must meet. Otherwise, the primer pair cannot be formed at all. For instance, the length of the primer is generally between 18 and 26 bases. The difference between the lengths of the up primer and down primer should be strictly less than 3, and so on. Soft constraints are a type of degenerate primer design rule that includes phrases such as "at most" and "at least." In the design of degenerate primers with maximum coverage, for example, the degeneracy is required to be d at most.

The ants use the roulette wheel instead of the node with the highest probability to choose the next node in their path and avoid falling into a local optimum. The simplified view of the construction graph model is shown in Figure 2.

At the construction stage, heuristic factors and pheromone factors associated with the constraints guide the ants toward selecting the next base or the next base combination in the primer sequence. The heuristic factors play a vital role, especially in guiding the choice of the next node. The probability state transition formula is shown in the following formula:

$$P_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{k \in \text{allowed}_k} [\tau_{ik}]^{\alpha} * [\eta_{ik}]^{\beta}}, & if\ j \in \text{allowed}_k, \\ 0, & \text{Otherwise}, \end{cases} \tag{2}$$

where i and j, respectively, represent the starting node; τ_{ij} is a pheromone factor, which mainly includes two operations: pheromone volatilization and pheromone enhancement. The pheromone update strategy used in this paper is as follows: after the kth ant completes a path search, the pheromone is updated for all paths on the route, and the pheromone increment is related to the overall route of this search, which is a global information update. The detailed update process of τ_{ij} is described in [22]; η_{ij} is a heuristic factor; the weights of the two factors η_{ij} and τ_{ij} are denoted by β and α, respectively; allowed k is a collection of nodes that ants have not accessed. After heuristic analysis, the following values can be set:

$$\eta_{ij} = \frac{\text{Converge}_{ij}}{\text{Degeneracy}_{ij}}, \tag{3}$$

where Degeneracy_{ij} denotes the degeneracy of edge arc ij and Converge_{ij} denotes the coverage of edge arc ij. As the study in this paper uses a maximum coverage degeneracy primer design, the degeneracy of primers is constant, and the greater coverage represents the better experimental results. Therefore, the heuristic factor η is larger, and the probability of the side arc being selected again is relatively large under the same conditions.

In order to meet different constraints, a local search process called LocalSearch is designed in this paper, including three ideas pretreatment before optimization, processing in the path construction process, and constraint processing after the end of path construction. Next, the situation will be introduced. First, some constraints need to be preprocessed before optimization. GC Clamp, for instance, can detect whether the two bases at the end of the $3'$end of primer S are G or C. However, considering another constraint condition, the bases should be distributed randomly; it means that polypurine or polypyrimidine should not appear, and no more than three consecutive G or C should occur at the $3'$end of primer S. Thus, three bases can be set at the end of the $3'$end of primers, and their combination range must be restricted. Second, some constraints need to be dealt with in the optimization process. For instance, the length of primers is generally between 18 and 26 bases, and this information was covered in detail in the process above on constructing solution paths, so it would not be repeated here. Third, some constraints need to be addressed near or after optimization. For example, the difference between the up primer's length and the down primer's length should be strictly less than 3. After designing the up primer, its length is assumed to be 21, and in the process of finding the down primers, the length range of the down primer is narrowed down to 18–24. This method significantly reduces the search space and enhances the efficiency of the process when dealing with thousands of DNA template sequences. Furthermore, after the final design of primer pairs has been determined, it is also essential to judge the hairpin structure and complementary sequence between primer pairs. If there is a hairpin structure or complementary sequence, the primer pairs must be adjusted further.

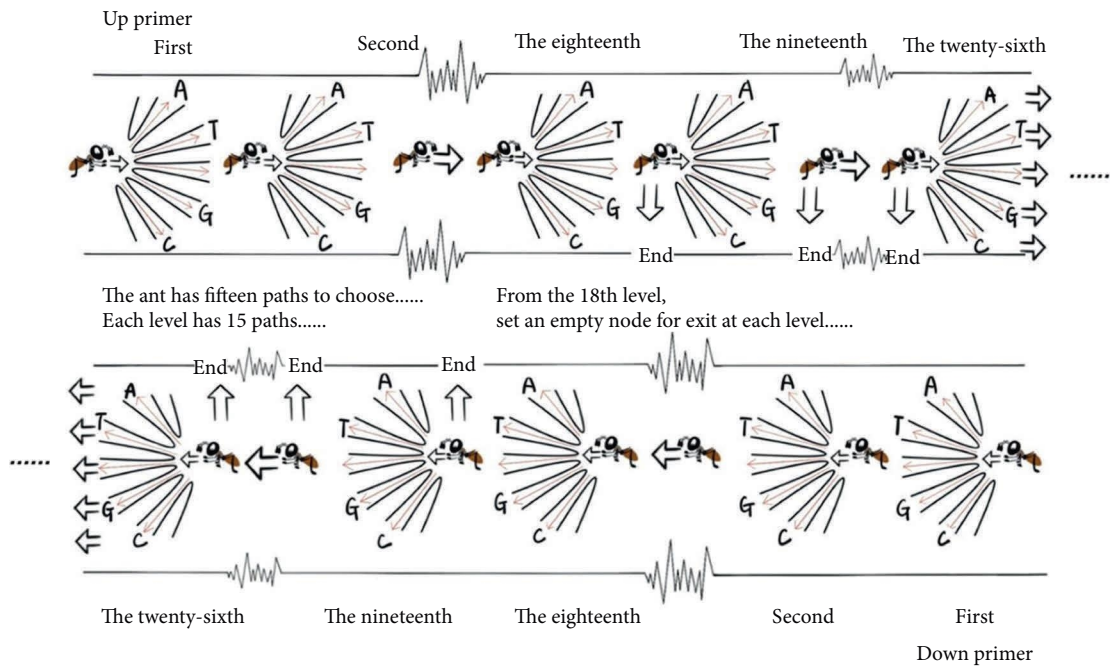


Figure 2: Simplified model of construction graph.

A degenerate primer evaluation is also a fundamental stage of ACOH. After a certain number of iterations, many ants generate solution paths, respectively, which involve comparing and analyzing multiple paths (actually primers). In order to express more clearly, this paper simplifies multiple paths into two paths for comparison. The evaluation of the solution path is analyzed and discussed in three cases: the two solution paths meet constraint conditions; one meets the constraint, the other does not; neither one meets the constraint. Next, the above three cases will be introduced, respectively. First, when both paths meet many constraints, they can be divided into two cases according to whether the coverage is the same. The soft constraint condition degeneracy can be used to assess when the coverage is the same. Eventually, the solution path with less degeneracy is taken. When the coverage is different, on the other hand, the result uses the solution path with greater coverage since experiments seek the design of degenerate primers with maximum coverage. Second, one of the two fits the constraint requirements, while the other does not by obviously choosing the solution path that meets the constraint conditions or meets more constraint conditions. Third, when both solutions fail to meet the constraint conditions, the final result discusses two cases based on whether the number of constraints violated by the two solution paths is the same. When two solution paths have the same amount of unsatisfied constraints, the solution chooses the one with greater coverage and then adjusts the primer pair to meet the constraints as much as possible. In contrast, when the number of constraints violated by two solution paths differs, one with fewer constraints violated by the two solution paths should obviously be selected. The specific execution process of the ACOH algorithm to solve the DPD problem is shown in Algorithm ACOH for Degenerate Primer Design pseudocode, which is as follows:

```
Algorithm ACOH for Degenerate Primer Design
Input
  f: the fitness evaluation function generated by Eq1
  Ω : constraint condition of degenerate primer design
Output
  S_bs : the optimal primer pairs
  Begin
  set parameters and initialize pheromone trails and heuristic information
  repeat
      for each ant k do
          Construct a complete primer pairs S generated by EQ2 and EQ3
              S ← LocalSearch (S)
          if f(S) ← f(S_bs) then
              S_bs ←S
          end if
      end for
      for each component i in graph do
          τ_ij ← update pheromone
      end for
  until maximum fitness evolution number reached
  return S_bs
```

ACOH algorithm complexity is as follows:

When constructing a complete primer pair, calculating the state transition probability formula requires $O(n * m)$ operations. In order to improve the quality of the constructed graph, $O(m)$ operations are necessary. It takes $O(n^2)$ operations to complete pheromone updates after each global path optimization. Therefore, the total time complexity is

$$T(n) = O\left(N_c * n^2 * m\right). \tag{4}$$

N_c, n, and m correspond to the number of iterations of the algorithm, the number of vertices in the graph, and the total number of ants, respectively. Thus, time complexity increases, with the size of the problem. The total complexity of space is

Table 1: α, β, ρ, mu, and phi five parameter details.

	α	B	ρ	mu	phi
General setup	2	8	0.02	(5, 10, 15, 20, 25)	(1, 2, 3, 4, 5)
Lower than general setup	Fall into local optimum	Solution set is simple	Reduced convergence rate		
Larger than general setup	Random search is reduced	Magnify the constraints and fall into local optimum	The preferred path is excluded		

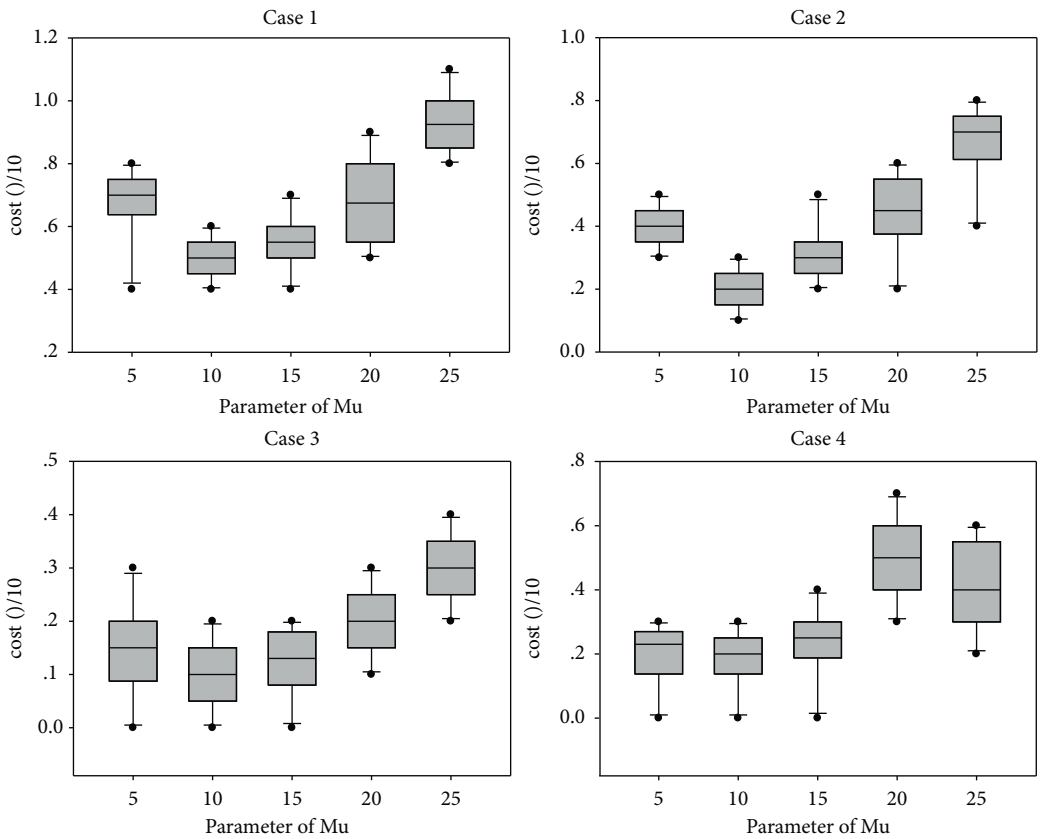


Figure 3: Setting parameter mu.

$$S(n) = O\left(n^2\right) + O(n * m). \tag{5}$$

3. Comparison and Experimental Analysis

100 sets of DNA template sequences for molecular biology are the experimental subjects of this paper. This algorithm experiment was carried out on a PC (3.40 GHz CPU and 16 GB RAM) using the eclipse. Java was selected as the programming language. The algorithms were run 30 times under all test cases independently, and the experimental results data were processed and analyzed for algorithms' comparison.

3.1. Parameter Adjustment. For designing degenerate primers, an ant colony optimization algorithm considers a number of parameters, including α, β, ρ, *mu*, and *phi*. The parameters are shown in Table 1 as follows:

A significant number of tests are required to determine the exact value of coverage mu, as well as the value of degeneracy phi, within a certain range. The final experimental results are depicted by a box-line plot, which shows that how many restrictions in the vertical axis cannot be satisfied for a given horizontal axis value, as can be seen in Figures 3 and 4.

By analyzing and comparing four typical data sets in Figure 3, it appears that at a parameter *mu* of 10, the boxplot distribution is more uniform, and the minimum value is the most optimal. In Case2, for instance, when $mu = 10$, the minimum, maximum, lower quartile, median, and upper quartile values are clearly better than the other values. In Case3, however, it has a superior upper quartile, median, and lower quartile than those in the two parametric cases, even though the minimum value at $mu = 10$ is the same as at $mu = 5$ or 10.

According to the analysis and comparison of the boxplots in Figure 4, there is an even distribution of the boxplot when the value of parameter phi is 3, and the minimum value at this time is the most optimal.

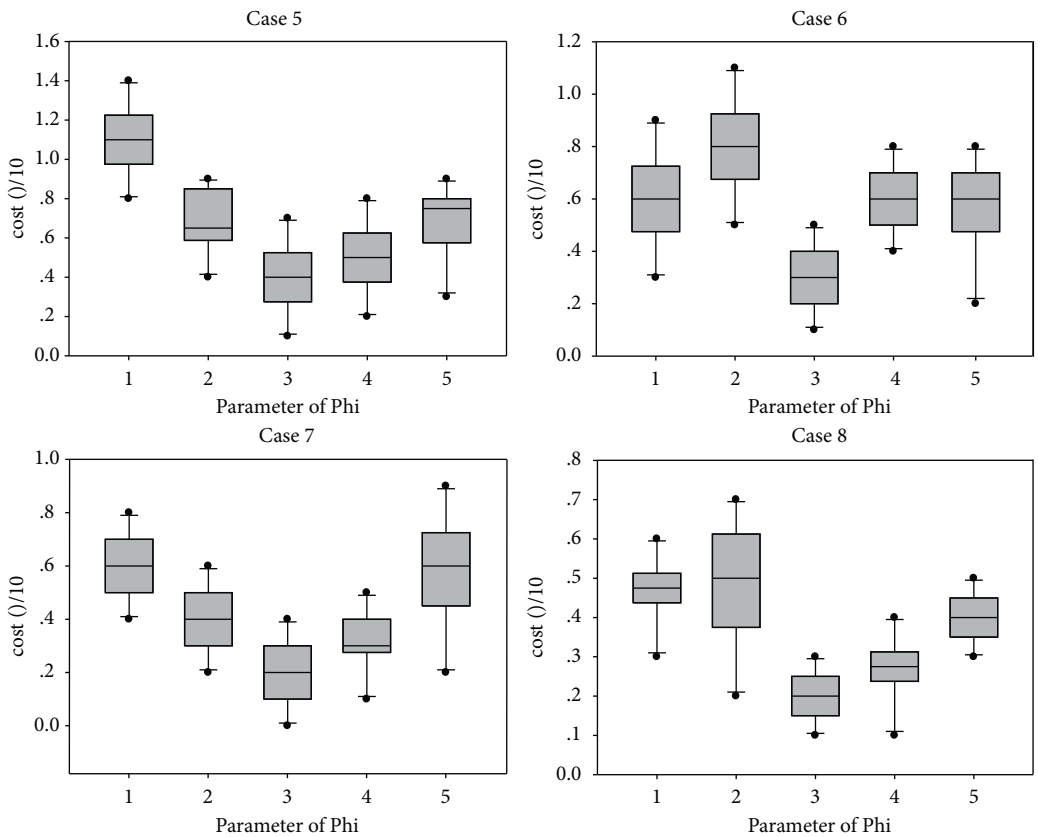


Figure 4: Setting parameter phi.

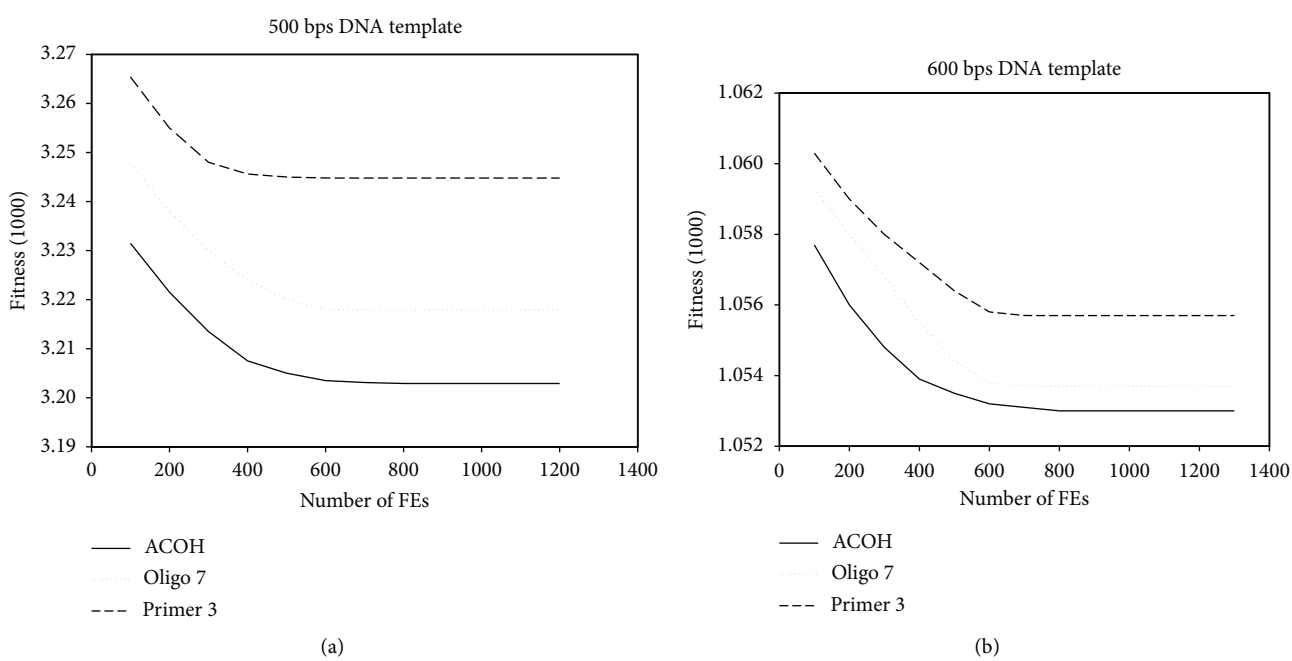


Figure 5: Convergence graphs for DPD. (a) 500 bps DNA template. (b) 600 bps DNA template.

3.2. Experimental Comparative Analysis. To make the experiment more convincing, the experiment results of degenerate primer design through the ant colony optimization algorithm are compared with those determined through Primer 3 and Oligo 7 [23]. The final test results need to be compared by two comparative analysis methods, i.e., descriptive statistical analysis and convergence analysis, to finally draw reliable conclusions.

Table 2: DNA sequence analysis at 500 bps and 600 bps under three conditions.

	ACOH	Primer 3	Oligo 7
500 bps up primer (3′- 5′)	CT[TA][TC][CG][TA] G[GT]C[CT][CG] [TA]A[CTA] [CG]CTA	[AC][AT]C[GC][ATC] [GA][AT]G[TC]C [TG]G[AT][AG]C [CG][AT][AT]TC	A[AG]CG[AT][GAT] AG[TC][CG]TG [AT]A[AC][GCT] [AT]G[TC]C
500 bps down primer (3′- 5′)	GAA[CA][CT][GT]	CT[TA][TA][GC]	C[CT]G[TA][TCG]
600 bps up primer (3′- 5′)	[CT][CT]G[TG]	C[GA][TA]G[GT]C[CT]	[CA]A[TA]GT
600 bps down primer (3′- 5′)	G[TG]ACGA[CGA] [TA]T[CA] GGA[TC]A[GA] GT[CA][CAT][CA] [TG][TA]C[CA] GAC[AG] GC[AG]CGG[TG] [TC]C[CAT]A [TA]G[TA][TC]G [GA]T[CA]	G[TA][AG][CTA] [CG]C[TA][CA] G[TAG]A[TC]A[GA] G[TGA][CA][CAT] [CA][TG][TA]C [CA][GA][AC][AG] GC[AG]C[AG]G [TG][TC]C[CAT] A[TA]G[TA][TC] [GA][GA][TA][CAG]	[GC][CT]GA[TAG] [TA]GC[GA]A [TG][AG]A[TC][TA] [GA]G[TA][CA] [CT][CGA]T[TA]C C[GA]ACA GC[CAG][CT]G [GA][TG]TC[CT] [GA][GTA]GAC [GA]G[TA][CA]

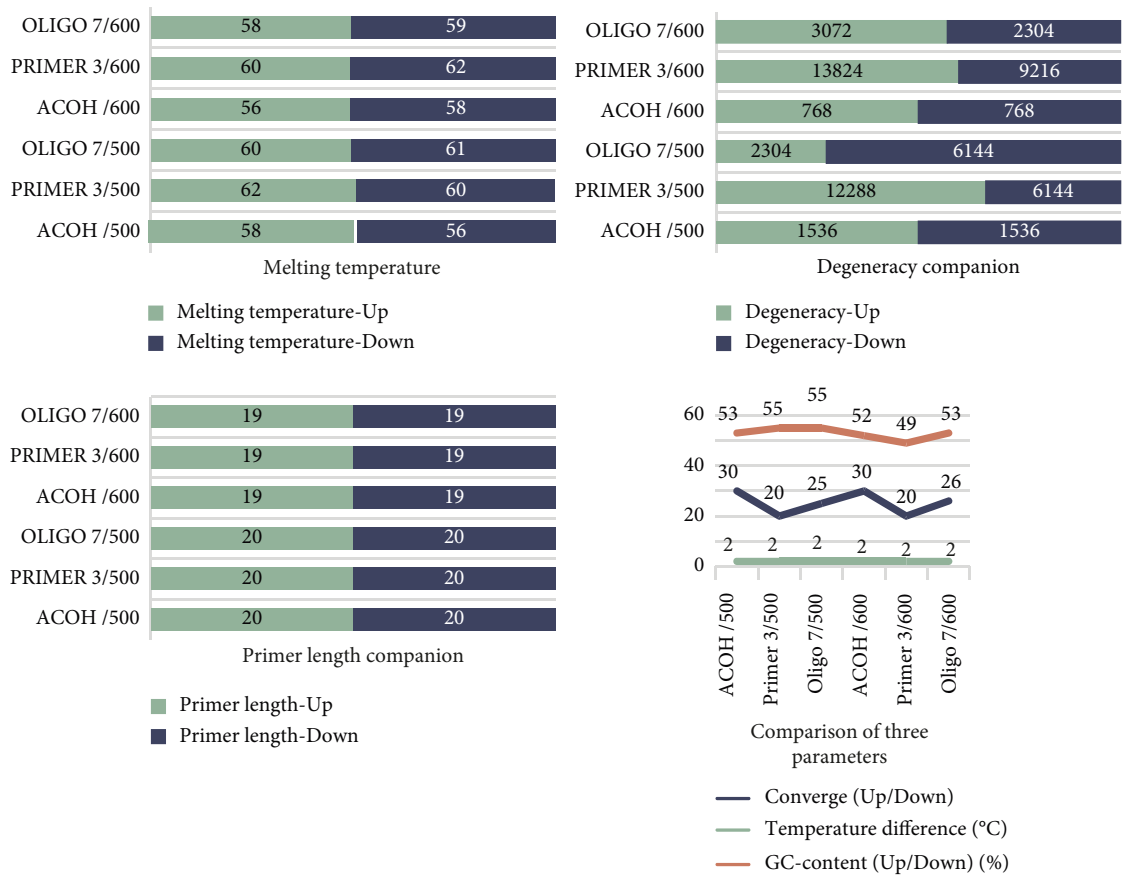


Figure 6: Comparison of four data at 500 bps and 600 bps.

3.2.1. Comparative Study Based on Convergence Analysis. The convergence analysis focuses on comparing the values of the fitness evaluation function of the standardized treatment for a certain number of fitness evaluations. ACOH, software Primer 3, and Oligo 7 solved the convergence curve of degenerate primer design on the DNA template of 500 bps and 600 bps, respectively, as shown in Figure 5.

As shown in Figure 5, on a 500 bps DNA template, Primer3 converged approximately 300 times with a minimum fitness evaluation function of 3.245. Oligo7 converged approximately 600 times with a minimum fitness evaluation function of 3.218. ACOH also converged approximately 600 times with a minimum fitness evaluation function of 3.204. Although Primer3 converges the earliest, it has the highest fitness evaluation function value. For the same number of

Table 3: An analysis of DNA sequences containing 100 bps, 200 bps, 300 bps, and 400 bps.

	100 bps	200 bps	300 bps	400 bps
Up primer (3′- 5′)	CG[TG]T[TA] [GA]G[GA]T[CA]C [GC]T[TG]C [GA]GT[GC]A	G[CT]A[GT] [TC]GG[CA]A[GT] [TG]CATT[CA] AG[TA]	G[GA]G[AG]AG [CT][GC]A[GA]A [GT]C[CT]GT[TA] [GTA]A[CA]	GG[GA]A[CA]C [AC]A[TC]ACG[TG] [CT][GA][GA]C [CGT]A[CT]
Down primer (3′- 5′)	GG[GT][GT]AA [GT][GT]CC[CT] ATG[TA]TG [CG]T[CA]	GT[GA][GT]TG [AG]A[CG]TGA[GA] [GT][GT]AG [GA]G	G[GA]G[AG]T[CTG] [GA][AC]G[CA]T [AC]C[TG]T [CA]A[GA]A	G[AG][GA]CGA [ACG]T[AC][AC][GA] G[CT]AA [CTA]T[AC]T

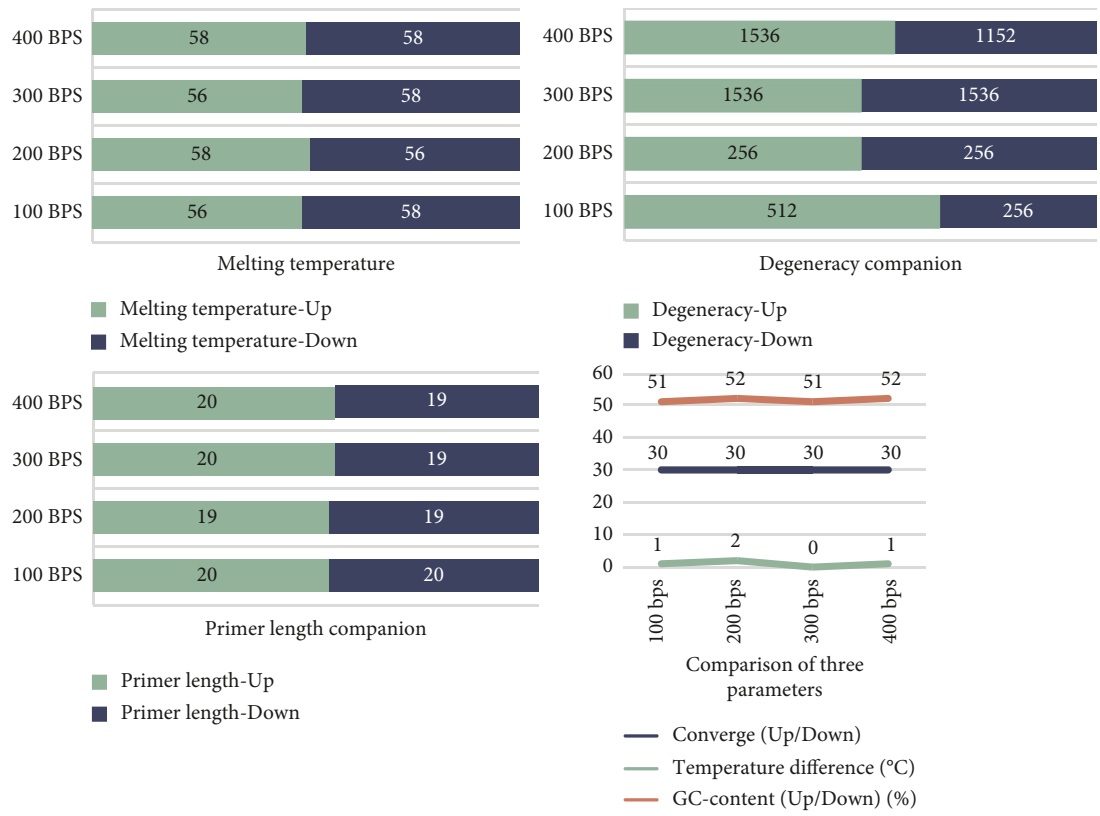


Figure 7: Comparison of four kinds of data at 100 bps, 200 bps, 300 bps, and 400 bps.

fitness evaluations, the normalized fitness evaluation function value of ACOH is the smallest among these three methods. ACOH has more advantages in this kind of problem since this section is to solve the minimum meeting constraint optimization problem. Similarly, ACOH has also shown an advantage on the DNA template of 600 bps.

3.2.2. Analysis and Comparison Based on Descriptive Statistical Methods. The up primer pair, the down primer pair, and many constraint parameters were compared at multiple angles. Table 2 and Figure 6 summarize the experimental results.

As can be seen from Table 2 and Figure 6, no matter 500 bps or 600 bps, the ACOH model performs best at degeneracy and convergence. In addition, at 500 bps, the data of up primer GC-clamp and down primer GC-clamp of the three models are [C] [C] [G] and [GG] [GG] [C]. At 600 bps, the up primer GC-clamp and down primer GC-clamp of the three conditions were [GG][G][CC] and [CG] [CG][GG]. Self-annealing and pair-annealing are no in all conditions. DPD solved by the ant colony optimization algorithm with the construction graph model is obviously lower than that solved by Primer 3 and Oligo 7. Moreover, the primer pair is specifically compared with Primer 3, which is inseparable from the preprocessing of the search space before constructing the solution path. Finally, only the condition optimized by ACOH has specificity.

Following that, four DNA template sequences, such as 100 bps, 200 bps, 300 bps, and 400 bps, will be used as test cases, with the corresponding primer pairs and constraint parameters acquired. As shown in Table 3 and Figure 7, DNA templates of 100 bps and 200 bps provide the same level of coverage as those of 300 bps and 400 bps, but their

degeneracy is relatively smaller. There is a correlation between the number of base pairs in the DNA sequence and the excellent search performance of the ant colony algorithm. In addition, the data of up primer GC-clamp and down primer GC-clamp of the four conditions are [GC] [G] [GG] [G]and [GG] [G] [G] [G]. Both self-annealing and pair-annealing were no in all models, but all models had specificity.

4. Conclusion

DPD is a hot topic in current research. Multiple limitations must be met by the candidate primers identified in this study. In addition, these primers should have a large amount of coverage and a small amount of degeneracy while allowing a minority of base mismatches. As a result, the research develops a DPD optimization model and proposes a solution to the problem based on the notion of ant colony optimization. Results of the experiments suggest that primers designed by this method have obvious effects on coverage, degeneracy, specificity, and stability and finally improve the solving efficiency. Besides, candidate primers are scored in this paper. Scoring is the process of weighing degeneracy and coverage into a single index, which necessitates the weight value before calculation. In practice, more pre-experiments are needed to determine the best weight values. As a result, in our next research, we will investigate how to create a multiobjective optimization model for the DPD and find an effective solution method. The solution method can give multiple candidate degenerate primer design schemes at the same time, which do not dominate each other.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

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

Acknowledgments

This work was supported by Liaoning Province Doctoral Research Start-up Fund Project (no. 2020-BS-152). This work was supported by Xingliao Talent Plan (no. XLYC1902095). This work was supported by Scientific Research Young Talents Project of Liaoning Education Department (no. LJKZ0266).

References

[1] P. Czechowski, L. J. Clarke, A. Cooper, and M. I. Stevens, "A primer to metabarcoding surveys of Antarctic terrestrial biodiversity," *Antarctic Science*, vol. 29, no. 1, pp. 3–15, 2017.

[2] H. Telenius, N. P. Carter, C. E. Bebb, M. Nordenskjo¨ld, B. A. Ponder, and A. Tunnacliffe, "Degenerate oligonucleotide-primed PCR: general amplification of target DNA by a single degenerate primer," *Genomics*, vol. 13, no. 3, pp. 718–725, 1992.

[3] L. W. Hugerth, H. A. Wefer, S. Lundin et al., "DegePrime, a program for degenerate primer design for Broad-Taxonomic-range PCR in Microbial Ecology Studies," *Applied and Environmental Microbiology*, vol. 80, no. 16, pp. 5116–5123, 2014.

[4] C. Linhart and R. Shamir, "The degenerate primer design problem: Theory and Applications," *Journal of Computational Biology*, vol. 12, no. 4, pp. 431–456, 2005.

[5] V. K. Singh, A. K. Mangalam, S. Dwivedi, and S. Naik, "Primer Premier: program for design of degenerate primers from a Protein sequence," *Biotechniques*, vol. 24, no. 2, pp. 318-319, 1998.

[6] C. Linhart and R. Shamir, "Degenerate primer design: theoretical analysis and the HYDEN program," *Methods in Molecular Biology*, vol. 402, no. 402, pp. 221–244, 2007.

[7] W. Treeratanajaru, S. Watcharamul, and L. Rajalida, "Degenerate primer design system for gene biodiversity study using dynamic pattern matching[C]," in *Proceedings of the 2012 7th International Symposium on Health Informatics and Bioinformatics*, pp. 102–106, IEEE, Nevsehir, Turkey, April 2012.

[8] S. Balla and S. Rajasekaran, "An efficient algorithm for minimum degeneracy primer selection," *IEEE Transactions on NanoBioscience*, vol. 6, no. 1, pp. 12–17, 2007.

[9] R. Souvenir, J. Buhler, G. Stormo, and W. Zhang, "Selecting degenerate multiplex PCR primers[C]," in *Proceedings of the International Workshop on Algorithms in Bioinformatics*, pp. 512–526, Springer, Berlin, Heidelberg, 2003.

[10] J. S. Wu, C. Lee, C. C. Wu, and Y. L. Shiue, "Primer design using genetic algorithm," *Bioinformatics*, vol. 20, no. 11, pp. 1710–1717, 2004.

[11] H. L. LiangC. Lee et al., "Multiplex PCR primer design for gene family using genetic algorithm[C]," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pp. 67–74, ACM, June 2005.

[12] Primer3web, "Primer3web," 2022, https://primer3.ut.ee/.

[13] Chang Bioscience, "Primo degenerate 3.2 help: degenerate PCR primer design," 2022, http://www.changbioscience.com/primo/dhowto.html.

[14] Codehop, "Codehop," 2022, http://blocks.fhcrc.org/codehop.html.

[15] C. Linhart and R. Shamir, "HYDEN–A software for designing degenerate primers," 2003, http://acgt.cs.tau.ac.il/hyden/.

[16] T. Cickovski, T. Flor, G. Irving-Sachs, P. Novikov, J. Parda, and G. Narasimhan, "GPUDePiCt: a Parallel Implementation of a Clustering algorithm for computing degenerate primers on Graphics processing Units," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 12, no. 2, pp. 445–454, 2015.

[17] H. Zhao, C. Zhang, X. Zheng, C. Zhang, and B. Zhang, "A decomposition-based many-objective ant colony optimization algorithm with adaptive solution construction and selection approaches," *Swarm and Evolutionary Computation*, vol. 68, Article ID 100977, 2022.

[18] Z. Liu, D. Jiang, C. Zhang, H. Zhao, Q. Zhao, and B. Zhang, "A Novel Fireworks algorithm for the Protein-Ligand Docking on the AutoDock," *Mobile Networks and Applications*, vol. 26, no. 2, pp. 657–668, 2021.

[19] L. Ma, M. Huang, S. Yang, R. Wang, and X. Wang, "An adaptive localized decision variable analysis approach to large-scale multiobjective and many-objective optimization [J]," *IEEE Transactions on Cybernetics*, vol. 52, pp. 6684–6696, 2021.

[20] L. Ma, X. Wang, L. Wang, Y. Shi, and M. Huang, "TCDA: Truthful combinatorial double auctions for mobile edge computing in industrial Internet of Things[J]," *IEEE Transactions on Mobile Computing*, p. 1, 2021.

[21] L. Ma, S. Cheng, and Y. Shi, "Enhancing Learning efficiency of Brain Storm optimization via Orthogonal Learning design," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 11, pp. 6723–6742, 2021.

[22] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[23] W. Rychlik, "OLIGO 7 primer analysis software[J]," *Methods in Molecular Biology (Clifton, NJ, USA)*, pp. 402: 35–60, 2007.

Hindawi
Discrete Dynamics in Nature and Society
Volume 2022, Article ID 1190174, 15 pages
https://doi.org/10.1155/2022/1190174

Research Article

A Hybrid Search Model for Constrained Optimization

Xiaoli Gao,[1] **Yangfei Yuan**,[2] **Jie Li**,[1] **and Weifeng Gao**[2]

[1]*Sichuan Jiuzhou Electric Appliance Group Co., Ltd., Mianyang, Sichuan 621000, China*
[2]*School of Mathematics and Statistics, Xidian University, Xi'an 710126, China*

Correspondence should be addressed to Xiaoli Gao; kewangdexin@126.com

Received 7 May 2022; Accepted 22 August 2022; Published 28 September 2022

Academic Editor: Shi Cheng

This paper proposes a hybrid model based on decomposition for constrained optimization problems. Firstly, a constrained optimization problem is transformed into a biobjective optimization problem. Then, the biobjective optimization problem is divided into a set of subproblems, and different subproblems are assigned to different Fitness functions by the direction vectors. Different from decomposition-based multiobjective optimization algorithms in which each subproblem is optimized by using the information of its neighboring subproblems, the neighbors of each subproblem are deFined based on corresponding direction vector only in the method. By combining three main components, namely, the local search model, the global search model, and the direction vector adjusting strategy, the population can gradually move toward the global optimal solution. Experiments on two sets of test problems and Five real-world engineering design problems have shown that the proposed method performs better than or is competitive with other compared methods.

1. Introduction

Constrained optimization has a wide application background in many important Fields, such as economics, engineering, and science [1–3]. In general, the mathematical deFinition of a constrained optimization problem (COP) is as below:

$$\min f(X), X = (x_1, \ldots, x_d) \in \Omega, l_j \le x_j \le u_j st: g_r(X) \le 0, \quad r = 1, \ldots, k h_w(X) = 0, \quad w = 1, \ldots, z, \tag{1}$$

where X represents a d-dimensional solution vector. $f(X)$ represents the objective function. $h_w(X)$ and $g_r(X)$ denote z-equality constraints and k-inequality constraints. x_j is restricted by the upper and lower bounds u_j and l_j, respectively.

In COPs, an equality constraint is generally converted into the following inequality forms.

$$|h_w(X)| - \xi \le 0, \quad w = 1, \ldots, z, \tag{2}$$

where ξ denotes a small positive value (i.e., 10^{-4}). In order to judge whether a solution X in COPs is a feasible solution, we must consider its overall constraint violation degree, which is computed as below:

$$G(X) = \sum_{r=1}^{k} \max\{0, g_r(X)\} + \sum_{w=1}^{z} \max\{0, |h_w(X) - \xi|\}, \tag{3}$$

where $G(X) \ge 0$ and X satisFies the constraints if and only if $G(X) = 0$.

Evolutionary algorithm (EA), which is a metaheuristic algorithm, has been adopted to deal with COPs in the past two decades. When EAs are employed to solve COPs, the constraint-handling techniques (CHTs) should be considered. The current popular CHTs include the penalty function methods [4–7], the multiobjective optimization methods [8–13], the feasibility rule methods [14–19], the ϵ constrained methods [20–22], and the hybrid methods [23–26]. In the penalty function methods, a penalty Fitness function is deFined by adding a penalty term to the objective function. In the feasibility rule methods, the feasible individuals are superior to the infeasible individuals. The ϵ constrained method is a representative CHT, in which the ϵ level is utilized to relax the constraints. And the hybrid method

solves COPs by combining multiple constraint-handling techniques.

The multiobjective optimization methods have been adopted to solve COPs in the last two decades. These methods always transform a COP into a biobjective optimization problem (BOP), in which one objective is the overall constraint violation degree $G(X)$ and another objective is the original objective $f(X)$. Then, the multiobjective optimization techniques, such as the Pareto dominance or the aggregation method, are utilized to compare the individuals. For example, Wang et al. [27] employed a dynamic hybrid model for solving COPs, in which Pareto dominance is employed for the comparison. Gao et al. [28] proposed a dual-population method to solve COPs, where $f(X)$ and $G(X)$ are optimized by the corresponding subpopulation, respectively. Moreover, Wang et al. [29] utilized the correlation between the objective function and the constraints to deal with COPs.

Decomposition method [30] is a representative multiobjective optimization method. To solve COPs through the decomposition-based multiobjective optimization methods, the transformed BOP is converted into a set of subproblems; that is, a group of Fitness functions are constructed by assigning different direction vectors between $f(X)$ and $G(X)$. Generally, in the decomposition-based multiobjective optimization method, each individual optimizes a subproblem by combining the information of its neighboring individuals. However, little effort to optimize each subproblem by using the information of its direction vector in the objective space.

Based on the above analysis, a hybrid search model for constrained optimization, called HyCO, is designed to solve COPs in this paper. First of all, a BOP is decomposed into K subproblems. Then, to balance the diversity and convergence, the local and global models are employed to optimize these subproblems. During the local search model, the whole population is decomposed into K subpopulations by adopting the classiFication operator, and each subpopulation optimizes a subproblem. During the global search model, the whole population is guided by a deFined search direction toimprove the convergence. In the process, differential evolution (DE) is utilized to generate the offsprings, and the direction vectors are adjusted to Fit the characteristic of COPs. Furthermore, a simple restart strategy is proposed by Wang et al. [31] to handle complex constraints. The performance of HyCO is tested on IEEE CEC 2010, IEEE CEC 2017, and Five engineering problems. The results show that HyCO is more competitive than other selected methods.

2. Multiobjective Optimization and Vector Angle

2.1. Multiobjective Optimization. Some details of multiobjective optimization problem (MOP) are introduced in this section. Generally speaking, a MOP can be expressed as

$$\min F(X) = \left(f_1(X), \ldots, f_p(X)\right) st\colon\ X \in S, \tag{4}$$

where $S = \prod_{i=1}^{n} [a_i, b_i]$ represents the n-dimensional decision space. $f_i(X)$ denotes the ith objective function. For two solutions X and Y, some concepts related to MOP are introduced as follows.

Definition 1. X is said to dominate Y (i.e., $X \preccurlyeq Y$). If $\forall i \in 1,..p, f_i(X) \le f_i(Y)$ and $\exists i \in 1,..p, f_i(X) < f_i(Y)$.

Definition 2. $X^* \in S$ is called a Pareto optimal solution. If $\exists X \in S$, such that $X \preccurlyeq X^*$.

Definition 3. A set of all Pareto optimal solutions is called the Pareto set (PS).

Definition 4. The Pareto front (PF) is the set of all Pareto optimal objective vectors (i.e., PF = $\{F(X) | X \in \text{PS}\}$).

2.2. Vector Angle. In MOPs, the vector angle represents the angle between two individuals in the objective space. Typically, for two individuals X_j and X_q, the vector angle between them can be computed as below:

$$\text{angle}\left(X_j, X_q\right) = \arccos\left(\frac{F*\left(\text{X}_j\right)^T F*\left(\text{X}_q\right)}{\left\|F*\left(\text{X}_j\right)\right\| \times \left\|\text{F}*\left(\text{X}_q\right)\right\|}\right), \tag{5}$$

where $F*(\text{X}_j) = (f_1^*(\text{X}_j), f_2^*(\text{X}_j), \ldots, f_p^*(\text{X}_j))$ is the jth individual's normalized objective vector, and $f_i^*(\text{X}_j)$ is computed according to the following equation:

$$f_i^*\left(\text{X}_j\right) = \frac{f_i\left(X_j\right) - Z_i^{\min}}{Z_i^{\max} - Z_i^{\min}}, \tag{6}$$

where $Z_i^{\min}$ and $Z_i^{\max}$ represent the minimum and maximum values of the ith objective. $f_i(X_j)$ represents the ith objective function value. $\|\cdot\|$ represents the norm of a vector. Generally, the vector angle is used to maintain the population diversity for MOPs [32, 33]. SpeciFically, if the vector angle of two individuals is small enough, then their search directions are similar. On the contrary, a large vector angle of two individuals means the different search directions, and the diversity between them can be maintained. Motivated by the above considerations, a new clustering method is designed in HyCO and the population (P) would be clustered into K subpopulations according to the vector angle to maintain the diversity.

3. Proposed Method

3.1. Motivation. When using EAs to solve COPs, there are two important issues need to be solved: firstly, achieving the balance between the diversity and convergence; secondly, achieving the balance between the constraints and objective function. In HyCO, the local and global search models based on decomposition are proposed to balance the diversity and convergence. Specifically, in the local search model, a clustering method is designed to divide the population into several subpopulations, and each subpopulation optimizes a subproblem to maintain the population diversity. In the global search model, a direction vector is deFined to guide the evolution and enhance the population convergence. In addition, a direction vector

Input: The population size m, the number of subpopulations K, and the total number of function evaluations T_FEs.
Output: The best feasible solutions in P.
Initialize a population randomly $P = \{X_1, \ldots, X_m\}$;
Calculate $f(X_i)$ and $G(X_i)$ of each individual X_i in P.
while stopping conditions are not satisfied do
  Execute **the local search model**;
  Execute **the global search model**;
  Execute **DVA**;
  Execute **the restart strategy**;
end while

Algorithm 1: HyCO.

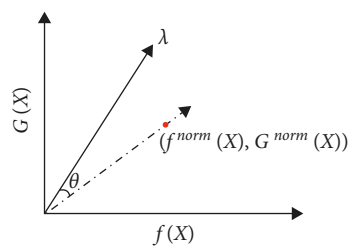


Figure 1: Illustration of the angle between the direction vector and the normalized objective vector.

Input: The population $P = \{X_1, \ldots, X_m\}$.
Output: K subpopulations $P = \{SubP_1, \ldots, SubP_k\}$.
Calculate the direction vectors $\{(\lambda_j, 1-\lambda_j)\}$ $(j = 1, \ldots, k)$ according to DVA;
for $j = 1$: K do
  Calculate the angle θ_{X_i,λ_j} according to (9);
  Find the corresponding minimum $[m/K]$ individuals, which are the minimum distance to the direction vector $(\lambda_j, 1-\lambda_j)$, to form a subpopulation;
  Eliminate these solutions from P;
end for

Algorithm 2: Classification operator.

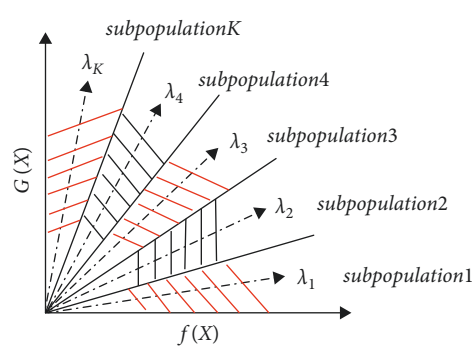


Figure 2: Illustration of the subpopulation assignment.

adjustment strategy (DVA) is used in HyCO to balance the objective and constraints, which can guide the population to converge to the feasible optimal solution.

Based on the above considerations, this paper utilizes the local and global search models based on decomposition to solve COPs.

Input: Entire population P^t.
Output: Updated subpopulation $\mathrm{SubP}_i^t\,(i = 1, \ldots, K)$.
Divide the population into K subpopulations $P^t = \{SubP_1^t, \ldots, SubP_K^t\}$ by Algorithm 2
Calculate the direction vectors $(\lambda_i^t, 1 - \lambda_i^t)\,(i = 1, \ldots, K)$ by Algorithm 7;
for $i = 1$: k do
  Set $\mathrm{SubP}_j^{t+1} = \varnothing$;
  Generate offspring $\mathrm{OSubP}_j^{t+1} = \{U_1^T, \ldots, U_{m/k}^t\}$ by Algorithm 4;
  Select the direction vector $(\lambda_i^t, 1 - \lambda_i^t)$ corresponding to $\mathrm{Sub}P_i^{t+1}$;
  for $i = 1$: m/K do
    if $g^{ws}(U_j^t|\lambda_i^t) < g^{ws}(X_j^t|\lambda_i^t)$ then
      $\mathrm{Sub}P_i^{t+1} = \mathrm{Sub}P_i^{t+1} \cup U_j^t$;
    else
      $\mathrm{Sub}P_i^{t+1} = \mathrm{Sub}P_i^{t+1} \cup X_j^t$;
    end if
  end for
end for

Algorithm 3: Local search model.

Input: $\mathrm{SubP}_i^t\,(i = 1, \ldots, K)$: Initial subpopulations.
Output: $\mathrm{OSubP}_i^t\,(i = 1, \ldots, K)$: Offspring subpopulations.
for $j = 1$: K do
  Set $\mathrm{OSubP}_j^t = \varnothing$;
  for $j = 1, \ldots, m/K$ do
    Randomly generate a F value in [0, 1];
    if $rand < 0.5$ then
      Generate a candidate solution U_i^t according to (10);
    else
      Generate a candidate solution U_i^t according to (11) and (12);
    end if
      $\mathrm{OSubP}_j^t = \mathrm{OSubP}_j^t \cup U_i^t$;
    end for
  end for

Algorithm 4: Local search algorithm.

Input: Initial population P^t.
Output: Updated population P^{t+1}.
Set $P^{t+1} = \varnothing$;
Calculate λ_c^t according to (13);
Generate an offspring population $OP^t = \{U_1^t, \ldots, U_m^t\}$ by Algorithm 6;
for $i = 1$: m do
  if $g^{ws}(U_i^t|\lambda_c^t) < g^{ws}(X_i^t|\lambda_c^t)$ then
    $P^{t+1} = P^{t+1} \cup U_i^t$;
  else
    $P^{t+1} = P^{t+1} \cup X_i^t$;
  end if
end for

Algorithm 5: Global search model.

3.2. HyCO. In HyCO, a transformed BOP is first decomposed into K subproblems. Next, the local and global search models are designed to optimize these subproblems, and DVA is proposed to adjust the direction vector. The detailed steps of HyCO are provided in Algorithm 1.

DE [34] is employed to generate the offsprings since its powerful search performance. Then, the weighted sum approach is adopted to compare the fitness of two candidate solutions. For a solution X_i^t, its weight sum can be deFined as below:

Input: Initial population P^t, Direction vector $(\lambda_c^t, 1-\lambda_c^t)$.
Output: Offspring population OP^t.
Set $\mathrm{OP}^t = \varnothing$;
for $i = 1$: m do
  Randomly generate an F value from $\{1.0, 0.8, \text{or } 0,6\}$;
  Randomly generate a CR value from $\{1.0, 0.2, \text{or}, 1\}$;
  if $rand < 0.5$ then
    Create a candidate solution U_i^t according to (14);
  else
    Create a candidate solution U_i^t according to (15);
  end if
    $\mathrm{OP}^t = \mathrm{OP}^t \cup \vec{u}_i^t$;
end for

Algorithm 6: Global search algorithm.

Set $\mathrm{DV} = \varnothing$;
if $t/T < 0.85$ then
  $\varepsilon = \varepsilon_0 (1 - t/T)^{cp}$, $cp = -(\log\varepsilon_0 + \beta/\log(1-\rho))$;
else
  $\varepsilon = 0$;
end if
Calculate the proportion of feasible solutions (pf) in P;
if $(t/T) \geq 0.85 \| pf \geq 0.85 \| G_{\min} \geq \varepsilon$ then
  $\xi = 10^{-18}$;
else
  $\xi = (1/1 + e^{\Gamma(t/T-\gamma)})$;
end if
for $i = 1$: K do
  $\lambda_i = (i/m) \cdot \xi$;
  $\mathrm{DV} = \mathrm{DV} \cup (\lambda_i, 1-\lambda_i)$;
end for

Algorithm 7: DVA.

Table 1: The population size m, the number of subpopulations K, and the total number of function evaluations T_FEs.

Test functions	T_FEs	m	K
18 COPs ($d = 10$) from CEC 2010	$2.0E+05$	80	14
18 COPs ($d = 30$) from CEC 2010	$6.0E+05$	100	15
28 COPs ($d = 50$) from CEC 2017	$1.0E+06$	100	15
28 COPs ($d = 100$) from CEC 2017	$2.0E+06$	100	16

$$g^{ws}\left(X_i^t | \lambda_j^t\right) = \lambda_j^t f^{\mathrm{norm}}\left(X_i^t\right) + \left(1 - \lambda_j^t\right) G^{\mathrm{norm}}\left(X_i^t\right), \tag{7}$$

where

$$f^{\mathrm{norm}}\left(X_i^t\right) = \frac{f\left(X_i^t\right) - f_{\min}^t}{f_{\max}^t - f_{\min}^t} G^{\mathrm{norm}}\left(X_i^t\right) = \frac{G\left(X_i^t\right) - G_{\min}^t}{G_{\max}^t - G_{\min}^t} \lambda_j^t = \frac{J}{K} \cdot \xi, \tag{8}$$

where ξ is a parameter in DVA. $f_{\min}^t$ and $f_{\max}^t$ represent the minimum and maximum objective function values, respectively. t represents the current generation number. $G_{\min}^t$ represents the minimum overall constraint violation. $(\lambda_j^t, 1-\lambda_j^t)$ is the direction vector. $G_{\max}^t$ represents the maximum overall constraint violation.

In the following sections, the local search model, the global search model, and DVA are introduced, respectively.

3.3. Local Search Model. The purpose of the local search model is to optimize each subproblem from different search directions and Find the promising search directions. In this model, the whole population is divided into K subpopulations by a classiFication operator and each subpopulation is used to optimize its assigned subproblem.

Table 2: Results of HyCO and other compared algorithms on 18 COPs with 10 d from CEC 2010.

Problem	Criteria	FROFI	ITLBO	DeCODE	AIS-IRP	ECHTDE	HyCO
C01	Mean $O \pm S$	−7.47E − 01 ± 1.35E − 03≈	−7.47E − 01 ± 1.87E − 03≈	−7.46E − 01 ± 5.02E − 03−	−7.47E − 01 ± 1.30E − 03≈	−7.47E − 01 ± 1.40E − 03≈	−7.47E − 01 ± 1.35E − 03
C02	Mean $O \pm S$	−2.02E + 00 ± 1.41E − 01−	−2.03E + 00 ± 8.14E − 02−	−2.18E + 00 ± 1.27E − 01+	−2.27E + 00 ± 2.00E − 03+	−2.27E + 00 ± 6.70E − 03+	−2.09E + 00 ± 2.53E − 01
C03	Mean $O \pm S$	0.00E + 00 ± 0.00E + 00+	0.00E + 00 ± 0.00E + 00+	0.00E + 00 ± 0.00E + 00+	0.00E + 00 ± 0.00E + 00+	0.00E + 00 ± 0.00E + 00+	7.10E + 00 ± 3.62E + 00
C04	Mean $O \pm S$	−1.00E − 05 ± 0.00E + 00≈	−1.00E − 05 ± 3.39E − 15≈	−1.00E − 05 ± 8.42E − 16≈	−9.97E − 06 ± 4.28E − 03-	−1.00E − 05 ± 0.00E + 00≈	−1.00E − 05 ± 1.55E − 15
C05	Mean $O \pm S$	−4.84E + 02 ± 8.09E − 07≈	−4.84E + 02 ± 1.11E − 11≈	−4.84E + 02 ± 3.48E − 13≈	−4.80E + 02 ± 6.30E + 00−	−4.11E + 02 ± 7.63E + 01−	−4.84E + 02 ± 3.48E − 13
C06	Mean $O \pm S$	−5.79E + 02 ± 5.04E − 04≈	−5.79E + 02 ± 2.39E − 04≈	−5.79E + 02 ± 1.29E − 13≈	−5.80E + 02 ± 7.30E − 08+	−5.62E + 02 ± 4.51E + 01−	−5.79E + 02 ± 1.35E − 13
C07	Mean $O \pm S$	0.00E + 00 ± 0.00E + 00≈	0.00E + 00 ± 0.00E + 00≈	0.00E + 00 ± 0.00E + 00≈	0.00E + 00 ± 0.00E + 00≈	1.33E − 01 ± 7.28E − 01−	0.00E + 00 ± 0.00E + 00
C08	Mean $O \pm S$	7.11E + 00 ± 4.79E + 00−	8.47E + 00 ± 4.09E + 00−	8.56E + 00 ± 4.26E + 00−	4.09E + 00 ± 1.46E + 00+	6.16E + 00 ± 6.45E + 00−	5.96E + 00 ± 5.30E + 00
C09	Mean $O \pm S$	2.50E + 01 ± 3.92E + 01−	0.00E + 00 ± 0.00E + 00+	4.91E + 00 ± 1.82E + 01−	2.70E + 01 ± 7.50E + 01−	1.47E − 01 ± 8.05E − 01+	1.76E + 00 ± 2.20E + 00
C10	Mean $O \pm S$	4.17E + 01 ± 8.69E − 06−	1.92E − 01 ± 9.62E − 01+	4.17E + 01 ± 2.20E − 14−	1.62E + 03 ± 5.00E + 02−	1.71E + 00 ± 7.66E + 00+	4.01E + 01 ± 8.53E + 00
C11	Mean $O \pm S$	−1.52E − 03 ± 5.63E − 14≈	−1.51E − 03 ± 1.30E − 05−	−1.52E − 03 ± 3.77E − 18≈	−9.20E − 04 ± 8.23E − 04−	−4.40E − 03 ± 1.57E − 02+	−1.52E − 03 ± 2.19E − 16
C12	Mean $O \pm S$	−3.84E + 02 ± 2.17E + 02+	−2.39E + 01 ± 1.14E + 02+	−1.99E + 00 ± 4.81E − 17−	−4.36E + 02 ± 6.02E + 01+	−1.72E + 02 ± 2.21E + 02+	−9.83E + 00 ± 3.15E + 01
C13	Mean $O \pm S$	−6.84E + 01 ± 2.52E − 09≈	−6.52E + 01 ± 1.78E + 00−	−6.84E+01 ± 2.90E − 14≈	−6.79E + 01 ± 3.11E − 01−	−6.51E + 01 ± 2.38E + 00−	−6.84E + 01 ± 8.30E − 02
C14	Mean $O \pm S$	0.00E + 00 ± 0.00E + 00≈	0.00E + 00 ± 0.00E + 00≈	0.00E + 00 ± 0.00E + 00≈	1.22E − 04 ± 2.90E − 08−	7.02E + 05 ± 3.19E + 06−	0.00E + 00 ± 0.00E + 00
C15	Mean $O \pm S$	3.09E + 00 ± 1.37E + 00+	3.54E + 00 ± 4.97E + 00−	2.94E + 00 ± 1.50E + 00+	0.00E + 00 ± 0.00E + 00+	2.34E + 13 ± 5.30E + 13−	3.38E + 00 ± 1.02E + 00
C16	Mean $O \pm S$	1.19E − 02 ± 2.07E − 02−	2.27E − 01 ± 3.11E − 01−	0.00E + 00 ± 0.00E + 00≈	0.00E + 00 ± 0.00E + 00≈	3.93E − 02 ± 4.28E − 02−	0.00E + 00 ± 0.00E + 00
C17	Mean $O \pm S$	7.83E − 02 ± 2.25E − 01−	3.91E − 01 ± 6.71E − 01−	2.05E − 11 ± 4.44E − 11≈	2.93E + 00 ± 2.29E + 00−	1.12E − 01 ± 3.32E − 01−	3.52E − 11 ± 3.88E − 11
C18	Mean $O \pm S$	5.23E − 26 ± 1.71E − 25≈	0.00E + 00 ± 0.00E + 00≈	0.00E + 00 ± 0.00E + 00≈	1.66E + 00 ± 1.27E + 00−	0.00E + 00 ± 0.00E + 00≈	0.00E + 00 ± 0.00E + 00
−		6	7	5	9	9	
+		3	4	3	6	6	
≈		9	3	10	3	3	

Table 3: Rankings obtained by the Friedman's test for HyCO and other compared algorithms on 18 COPs with 10 d from CEC 2010.

Algorithm	Ranking
HyCO	**3.25**
DeCODE	3.3333
AIS-IRP	3.3611
FROFI	3.4722
ITLBO	3.6667
ECHTDE	3.9167

Table 4: Results of HyCO and other compared algorithms by the multiple-problem Wilcoxon's test on 18 COPs with 10 d from CEC 2010.

HyCO vs.	R^+	R^-	ρ	$\alpha = 0.1$	$\alpha = 0.05$
FROFI	106.0	65.0	≥0.2	No	No
ITLBO	82.5	70.5	≥0.2	No	No
DeCODE	87.5	66.0	≥0.2	No	No
AIS-IRP	87.0	66.0	≥0.2	No	No
ECHTDE	93.0	59.5	≥0.2	No	No

To introduce the classification operator, the vector angle between the direction vector and the normalized objective vector is firstly defined as shown in Figure 1 and calculated as follows:

$$\theta_{X_i,\lambda_j} = \arccos \frac{F(X_i)\cdot w(\lambda_j)}{\left\|F(X_i)\right\| \times \left\|w(\lambda_j)\right\|}, \quad i = 1,\ldots,m. j = 1,\ldots,k, \tag{9}$$

where $F(X_i) = (f^{\text{norm}}(X_i), G^{\text{norm}}(X_i))$ is the ith individual's normalized objective vector. $w(\lambda_j) = (\lambda_j, 1-\lambda_j)$ is the direction vector, and $F(X_i)\cdot w(\lambda_j)$ denotes the inner product between $F(X_i)$ and $w(\lambda_j)$. Then, the classification operator is given in Algorithm 2.

As shown in Figure 2, each subpopulation owns a region which centered by a direction vector. Note that the Kth subpopulation contains all the remaining individuals in P, and the size of each subpopulation may not be equal. When the direction vector is adjusted, all individuals will be reclassified by the classification operator. Therefore, the individuals may be classified to the subpopulations different from the previous one, which results in the coevolution of different subpopulations. The whole process of the local search model is given in Algorithm 3.

In the process of local search, two modiFied DEs are employed to generate the offsprings. Their formulations are given as follows:

(i) DE/ModiFied/1

$$U_i = X_i + F\cdot\left(X_{best} - l\frac{(X_{mean}+X_i)}{2}\right) + F\cdot(X_{r1} - X_{r2}). \tag{10}$$

(ii) DE/ModiFied/2

$$\begin{aligned} U_i &= X_i + F\cdot(X_{r1}-X_i) + F\cdot(X_{r2}-X_{r3}), \quad if \quad f(X_{r1}) < f(X_i), \\ U_{i,j} &= \begin{matrix} X_{i,j}, & if\ rand_1 < rand_2 \\ X_{r1,j+m}\cdot H_{i,j}, & otherwise \end{matrix}, \quad j = 1,\ldots,d, \quad if \quad f(X_{r1}) \geq f(X_i), \\ H_{i,j} &= \begin{matrix} X_{r2,j} - X_{r3,j}, & if \quad g^{ws}(X_{r2}|\lambda_i) < g^{ws}(X_{r3}|\lambda_i) \\ X_{r3,j} - X_{r2,j}, & if \quad g^{ws}(X3|\lambda_i) < g^{ws}(X_{r2}|\lambda_i) \end{matrix}, \end{aligned} \tag{11}$$

where X_i and U_i represent the ith target vector and trial vector, respectively. X_{ij} and U_{ij} represent the jth dimension of them. X_{best} and X_{mean} denote the best individual and the mean vector in the subpopulation, respectively. X_{r1}, X_{r2}, and X_{r3} represent three individuals in P, which satisfy $X_{r1} \neq X_{r2} \neq X_{r3} \neq X_i$. $X_{r1,j}$, $X_{r2,j}$, and $X_{r3,j}$ represent the jth dimension of X_{r1}, X_{r2}, and X_{r3}, respectively. l is a random value in $[-1, 1]$. l is an integer selected from $\{1, 2\}$. F denotes the scaling factor. $\{\lambda_i, 1-\lambda_i\}$ represents the ith direction vector. $rand_1$ and $rand_2$ are randomly generated from $[0, 1]$.

As shown in (10), each subpopulation is guided by its best individual, which prevents the population from trapping into the local optimal solution. As shown in (11) and (12), the information of the individual with the smaller weighted sum is employed to generate the candidate solutions, which can enhance the rate of convergence. The procedure of the local search algorithm is described in Algorithm 4.

3.4. Global Search Model. In the local search model, each subproblem is optimized by corresponding direction vector, which may lead to the slow convergence rate. Candidate solutions are generated by using the individuals within one subpopulation, resulting in the weak information exchange among different subpopulations. Therefore, the diversity can be maintained but the convergence cannot be proved in the local search process. In order to improve the convergence,

Table 5: Results of HyCO and other compared algorithms on 18 COPs with 30 d from CEC 2010.

Problem	Criteria	FROFI	ITLBO	DeCODE	AIS-IRP	ECHTDE	HyCO
C01	Mean $O \pm S$	−8.21E − 01 ± 2.36E − 03+	−8.20E − 01 ± 8.95E − 04≈	−8.19E − 01 ± 3.20E − 03≈	−8.20E − 01 ± 3.25E − 04≈	−8.00E − 01 ± 1.79E − 02−	−8.19E − 01 ± 2.83E − 03
C02	Mean $O \pm S$	−2.00E + 00 ± 4.35E − 02−	−2.03E + 00 ± 7.64E − 02−	−2.23E + 00 ± 3.49E − 02−	−2.21E + 00 ± 2.84E − 03−	−1.99E + 00 ± 2.10E − 01−	−2.27E + 00 ± 1.04E − 02
C03	Mean $O \pm S$	2.87E + 01 ± 6.24E-08−	7.84E + 01 ± 6.31E + 01−	2.06E + 01 ± 1.31E + 01+	6.68E + 01 ± 4.26E + 02−	9.89E + 01 ± 6.26E + 01−	2.52E + 01 ± 9.51E + 00
C04	Mean $O \pm S$	−3.33E − 06 ± 4.13E − 10≈	1.69E − 03 ± 1.14E − 03−	−3.33E − 06 ± 6.92E − 12≈	1.98E − 03 ± 1.61E − 03−	−1.03E − 06 ± 9.01E − 03−	−3.33E − 06 ± 1.84E − 09
C05	Mean $O \pm S$	−4.81E + 02 ± 2.84E + 00−	−4.82E + 02 ± 1.73E + 00−	−4.83E + 02 ± 1.53E − 01−	−4.36E + 02 ± 2.51E + 01−	−1.06E + 02 ± 1.67E + 02−	−4.84E + 02 ± 6.11E − 02
C06	Mean $O \pm S$	−5.29E + 02 ± 5.71E − 01≈	−5.30E + 02 ± 4.80E − 01≈	−5.28E + 02 ± 1.46E + 00≈	−4.54E + 02 ± 4.79E + 01−	−1.38E + 02 ± 9.89E + 01−	−5.30E + 02 ± 1.52E − 01
C07	Mean $O \pm S$	0.00E + 00 ± 0.00E + 00≈	1.59E − 01 ± 7.97E − 01−	0.00E + 00 ± 0.00E + 00≈	1.07E + 00 ± 1.61E + 00−	1.33E − 01 ± 7.28E − 01−	1.91E − 27 ± 7.27E − 27
C08	Mean $O \pm S$	0.00E + 00 ± 0.00E + 00≈	1.14E + 01 ± 2.79E + 01−	0.00E + 00 ± 0.00E + 00≈	1.65E + 00 ± 6.41E − 01−	3.36E + 01 ± 1.11E + 02−	1.65E − 27 ± 4.56E − 27
C09	Mean $O \pm S$	4.30E + 01 ± 3.27E + 01−	2.86E + 00 ± 1.43E + 01−	8.97E + 00 ± 2.32E + 01−	1.57E + 00 ± 1.96E + 00−	4.24E + 01 ± 1.38E + 02−	1.76E − 01 ± 8.80E − 01
C10	Mean $O \pm S$	3.13E + 01 ± 8.22E − 02≈	3.29E + 01 ± 1.41E + 01−	3.13E + 01 ± 1.72E − 05≈	1.78E + 01 ± 1.88E + 01+	5.34E + 01 ± 8.83E + 01−	3.13E + 01 ± 1.68E − 06
C11	Mean $O \pm S$	−3.92E − 04 ± 2.64E − 06≈	−3.86E − 04 ± 1.14E − 05−	−3.92E − 04 ± 3.11E − 10≈	−1.58E − 04 ± 4.67E − 05−	2.60E − 03 ± 6.00E − 03▽−	−3.92E − 04 ± 6.11E − 10
C12	Mean $O \pm S$	−1.99E − 01 ± 1.42E − 06≈	−1.98E − 01 ± 2.39E − 03−	−1.99E − 01 ± 1.23E − 06≈	4.29E − 06 ± 4.52E − 04−	−2.51E + 01 ± 1.37E + 02▽−	-1.99E − 01 ± 2.66E-08
C13	Mean $O \pm S$	−6.83E + 01 ± 1.95E − 01+	−5.05E + 01 ± 1.18E + 00−	−6.73E + 01 ± 1.60E + 00+	−6.62E + 01 ± 2.27E − 01+	−6.46E + 01 ± 1.67E + 00+	−6.36E + 01 ± 1.93E + 00
C14	Mean $O \pm S$	9.80E − 29 ± 4.90E − 28≈	4.78E − 01 ± 1.32E + 00−	0.00E + 00 ± 0.00E + 00≈	8.68E − 07 ± 3.14E − 07−	1.24E + 05 ± 6.77E + 05−	1.29E − 27 ± 3.88E − 27
C15	Mean $O \pm S$	2.16E + 01 ± 8.03E-05≈	2.38E + 01 ± 2.51E + 01−	2.18E + 01 ± 1.14E + 00−	3.41E + 01 ± 3.82E + 01−	1.94E + 11 ± 4.35E + 11−	2.16E + 01 ± 1.87E-07
C16	Mean $O \pm S$	0.00E + 00 ± 0.00E + 00≈	0.00E + 00 ± 0.00E + 00≈	0.00E + 00 ± 0.00E + 00≈	8.21E − 02 ± 1.12E − 01−	0.00E + 00 ± 0.00E + 00≈	0.00E + 00 ± 0.00E + 00
C17	Mean $O \pm S$	1.59E − 01 ± 3.82E − 01−	9.65E − 01 ± 1.73E + 00−	4.48E − 02 ± 1.21E − 01−	3.61E + 00 ± 2.54E + 00−	2.75E − 01 ± 3.78E − 01−	4.97E − 12 ± 1.87E − 11
C18	Mean $O \pm S$	4.87E − 01 ± 1.25E + 00−	9.07E − 17 ± 3.18E − 16≈	3.03E − 06 ± 1.29E − 05−	4.02E + 01 ± 1.80E + 01−	0.00E + 00 ± 0.00E + 00≈	7.14E − 32 ± 1.02E − 31
−		6	14	6	15	15	
+		2	0	2	2	1	
≈		10	4	10	1	2	

Table 6: Rankings obtained by the Friedman's test for HyCO and other compared algorithms on 18 COPs with 30 d from CEC 2010.

Algorithm	Ranking
HyCO	**2.3056**
DeCODE	2.6389
FROFI	2.9444
ITLBO	4
AIS-IRP	4.3333
ECHTDE	4.7778

Table 7: Results of HyCO and other compared algorithms by the multiple-problem Wilcoxon test on 18 COPs with 30 d from CEC 2010.

HyCO vs.	R^+	R^-	ρ	$\alpha = 0.1$	$\alpha = 0.05$
FROFI	119.0	42.0	7.24E − 02	Yes	No
ITLBO	164.0	7.0	5.42E − 04	Yes	Yes
DeCODE	103.5	67.5	≥0.2	No	No
AIS-IRP	138.0	28.0	1.15E − 02	Yes	Yes
ECHTDE	134.0	19.0	6.04E − 03	Yes	Yes

the global search model is proposed. In this model, a direction vector is deFined to guide the whole population as follows:

$$\lambda_c = \frac{\sum_{i=1}^{a} \lambda_i}{a}, \tag{12}$$

where $(\lambda_i, 1 - \lambda_i)$ is the direction vector that a subproblem has been improved. a is the number of improved subproblems. The framework of the global search model is described in Algorithm 5.

In the process of global search, two DE operators are combined to generate the offsprings. Their formulations are introduced as follows [30, 35, 36]:

(i) DE/rand-to-best/1/bin

$$V_i = X_{r1} + F \cdot (X_{\text{best}} - X_{r1}) + F \cdot (X_{r2} - X_{r3})$$
$$U_{i,j} = \begin{cases} V_{i,j}, & \text{if} r\text{and}_j < \text{CR or} j = j_{\text{rand}}, \\ X_{i,j}, & \text{otherwise}, \end{cases} \tag{13}$$

(ii) DE/current-to-rand/1

$$U_i = X_i + r\text{and} \cdot (X_{r1} - X_i) + F \cdot (X_{r2} - X_{r3}), \tag{14}$$

where V_i represents the ith mutant vector. $V_{i,j}$ denotes the jth dimension of it. $r1$, $r2$, and $r3$ are three integers in P, which satisfy $r1 \neq r2 \neq r3 \neq i$. X_{best} is the best individual according to the weighted sum. CR denotes the crossover probability. And j_{rand} is a random value in $\{1, \ldots, d\}$.

With respect to (14), the solution X_{best} is utilized for enhancing the convergence. In (15), a randomly chosen solution X_{r1} is employed for promoting the diversity. In this paper, these two operations are executed with a probability of 0.5. Its effectiveness has been validated in [22, 37]. The whole process of the global search algorithm is introduced in Algorithm 6.

3.5. DVA. DVA is the major component when solving the transformed BOP through the local and global search models based on decomposition. For MOPs, the image of all Pareto optimal solutions is distributed on the PF [38]. However, for a BOP, only one global optimal solution needs to be obtained. Therefore, the direction vectors need to be adjusted to Fit the characteristic of COPs. DVA is proposed by Wang et al. [30], and the direction vector is adjusted according to the sigmoid function as follows:

$$\xi = \frac{1}{1 + e^{\Gamma(t/T - \gamma)}}, \tag{15}$$

where T represents the maximum generation number. γ and Γ are two positive values to control the change trend of ξ. Moreover, the ε constrained method is proposed to determine whether ξ should be reduced to a small number for COPs. The details of DVA are given in Algorithm 7.

4. Results and Discussion

4.1. Experiment Settings. To test the performance of HyCO, two sets of COPs are adopted. The first set contains 36 COPs from IEEE CEC 2010 [39] and the second set involves 56 COPs from IEEE CEC 2017 [39]. They have different characteristics, such as multimodality, extremely strong nonlinearity, rotated landscape, and so on. The population size (m), the number of subpopulations (K), and the total number of function evaluations (T_FEs), which are reported in Table 1, where d represents the dimension of COPs. In addition, each COP runs 25 times independently. μ in the restart strategy is set to 10^{-4}. ρ and β in the ϵ constrained methods are set to 0.85 and 6, respectively. In (15), Γ and γ are set to 30 and 0.75, respectively.

4.2. Experiments on the 36 COPs from IEEE CEC 2010. First of all, 36 COPs from CEC 2010 are tested in this section. Five competitive methods are selected: FROFI [40], ITLBO [41], DeCODE [30], AIS-IRP [42], and ECHTDE [43]. The experimental results of these methods are obtained from the literature [30, 37]. Since the true optimal value of this test suite is unknown, the "Mean O" and "S" are selected as the comparison criterion. "Mean O" and "S" are the mean and standard deviation of the results, respectively. Furthermore, the multiple-problem Wilcoxon's test and the Friedman's test are obtained via KEEL software [44].

In the case of COPs with 10d, the results of "Mean O" and "S," the Friedman's test, and the multiple-problem Wilcoxon's test are given in Tables 2–4, respectively. In Table 2, "▽" represents any feasible solutions of the compared algorithm cannot be found after the evolution. "+," "≈," and "−" represent that HyCO is worse than, competitive with, and better than the selected method, respectively. It can be seen from Table 2 that HyCO surpasses FROFI, ITLBO, DeCODE, AIS-IRP, and ECHTDE on 6, 7, 5, 9, and 9 test problems, respectively. In contrast, FROFI, ITLBO, DeCODE, AIS-IRP, and ECHTDE outperform HyCO on 3, 4, 3, 6, and 6 test problems, respectively. As shown in Table 4, the R^- values cannot exceed the R^+ values in all comparisons.

Table 8: Results of HyCO and other compared algorithms on 56 COPs from CEC 2017.

Problem	LSHADE44 (50 d)	UDE (50 d)	HyCO (50 d)	LSHADE44 (100 d)	UDE (100 d)	HyCO (100 d)
C01	0.00E + 00 ± 0.00E + 00≈	3.18E − 11 ± 7.74E − 11≈	1.65E − 21 ± 2.83E − 21	0.00E + 00 ± 0.00E + 00≈	1.79E − 03 ± 7.14E − 04−	2.08E − 09 ± 1.67E − 09
C02	0.00E + 00 ± 0.00E + 00≈	1.60E − 11 ± 2.71E − 11≈	2.61E − 21 ± 3.86E − 21	0.00E + 00 ± 0.00E + 00≈	1.56E − 03 ± 9.54E − 04−	1.71E − 09 ± 1.72E − 09
C03	8.95E + 05 ± 7.40E + 05−	1.09E + 02 ± 4.27E + 01−	1.38E − 20 ± 1.33E − 20	2.73E + 06 ± 9.66E + 05−	7.42E + 02 ± 2.17E + 02−	2.49E − 09 ± 1.67E − 09
C04	1.36E + 01 ± 5.44E − 15+	1.47E + 02 ± 2.55E + 01−	9.73E + 01 ± 2.95E + 01	1.37E+02 ± 4.62E − 01+	4.01E + 02 ± 4.96E + 01−	2.10E + 02 ± 3.69E + 01
C05	0.00E + 00 ± 0.00E + 00+	1.34E + 01 ± 3.35E + 00−	1.59E − 01 ± 7.97E − 01	3.28E − 05 ± 9.25E − 05+	7.54E + 01 ± 2.56E + 01−	3.19E − 01 ± 1.10E + 00
C06	7.51E + 03 ± 1.42E + 03−	7.43E + 02 ± 2.81E + 02−	1.87E + 02 ± 1.52E + 02	1.55E + 04 ± 1.59E + 03−	2.53E + 03 ± 5.31E + 02−	3.27E + 02 ± 9.75E + 01
C07	−1.79E + 02 ± 8.97E + 01+	−9.78E + 02 ± 2.13E + 02+	−1.10E + 02 ± 1.70E + 02	−3.02E + 02 ± 1.35E + 02+	−1.64E + 03 ± 3.29E + 02+	−3.48E + 01 ± 1.91E + 02
C08	−1.30E − 04 ± 2.77E − 20+	1.45E − 04 ± 1.41E − 04−	−3.81E − 05 ± 7.41E − 05	−4.81E − 05 ± 1.33E − 07+	2.97E − 03 ± 8.70E − 04▽+	▽
C09	−2.04E − 03 ± 1.33E − 18≈	−2.04E − 03 ± 3.87E − 16≈	−2.03E − 03 ± 8.63E − 06	−1.43E − 03 ± 2.21E − 19−	2.46E − 01 ± 3.85E − 01▽−	2.32E − 08 ± 4.57E − 08
C10	−4.83E − 05 ± 0.00E + 00≈	3.04E − 05 ± 3.61E − 05−	−2.81E − 05 ± 2.21E − 05	−1.72E − 05 ± 1.29E − 08+	5.57E − 04 ± 6.87E − 05≈	5.90E − 04 ± 8.64E − 05
C11	−1.77E + 00 ± 3.33E − 01+	▽≈	▽	▽≈	▽≈	▽
C12	4.98E + 01 ± 1.99E + 01−	2.09E + 01 ± 1.07E + 01−	1.54E + 01 ± 6.92E + 00	3.25E + 01 ± 8.19E − 01−	1.07E + 01 ± 1.00E + 01−	8.37E + 00 ± 5.04E + 00
C13	2.67E + 01 ± 1.36E + 01−	1.12E + 03 ± 5.55E + 02−	4.13E + 00 ± 1.99E + 01	8.07E + 01 ± 7.37E + 00+	▽−	1.08E + 02 ± 9.69E + 01
C14	1.40E + 00 ± 3.74E − 02−	1.23E + 00 ± 1.21E − 01−	1.10E + 00 ± 6.80E − 16	9.72E − 01 ± 1.94E − 02−	9.14E − 01 ± 7.97E − 02−	7.84E − 01 ± 3.95E − 16
C15	1.78E + 01 ± 3.00E + 00−	1.05E + 01 ± 1.57E + 00−	8.14E + 00 ± 2.16E + 00	1.81E + 01 ± 1.28E + 00−	1.80E + 01 ± 1.57E + 00−	1.18E+01 ± 1.87E − 15
C16	2.53E + 02 ± 1.62E + 01−	1.21E + 01 ± 1.54E + 00−	6.28E+00 ± 3.02E − 06	5.35E + 02 ± 3.08E + 01−	3.37E + 01 ± 7.88E + 00−	6.28E + 00 ± 0.00E + 00
C20	3.20E + 00 ± 1.43E − 01+	7.59E + 00 ± 2.05E + 00−	4.93E + 00 ± 8.18E − 01	9.36E + 00 ± 3.78E − 01+	1.89E + 01 ± 2.92E + 00−	1.01E + 01 ± 1.87E + 00
C21	6.29E + 01 ± 1.59E + 00−	6.43E + 00 ± 4.24E + 00−	1.62E + 01 ± 1.42E + 01	3.16E + 01 ± 2.99E − 03−	1.48E + 01 ± 1.21E + 01−	1.36E + 01 ± 9.18E + 00
C22	▽−	2.90E + 03 ± 1.61E + 03▽−	8.71E + 02 ± 4.65E + 02	▽−	▽−	9.14E + 02 ± 4.43E + 02
C23	1.33E + 00 ± 6.16E − 02−	1.10E + 00 ± 1.11E − 02+	1.17E + 00 ± 1.35E − 01	9.69E − 01 ± 4.26E − 02−	7.85E − 01 ± 5.25E − 03+	8.13E − 01 ± 5.55E − 02
C24	1.43E + 01 ± 1.28E + 00−	1.13E + 01 ± 1.76E + 00−	4.87E + 00 ± 1.28E + 00	1.71E + 01 ± 1.43E + 00−	1.81E + 01 ± 7.25E − 15−	8.64E + 00 ± 2.59E − 06
C25	2.48E + 02 ± 1.58E + 01−	2.24E + 01 ± 6.01E + 00−	6.79 E + 00 ± 1.74 E + 00	5.44 E + 02 ± 2.86 E + 01−	1.65 E + 02 ± 3.81 E + 01−	7.79 E + 01 ± 2.83 E + 01
−	12	16		12	17	
+	6	2		7	3	
≈	4	4		3	2	

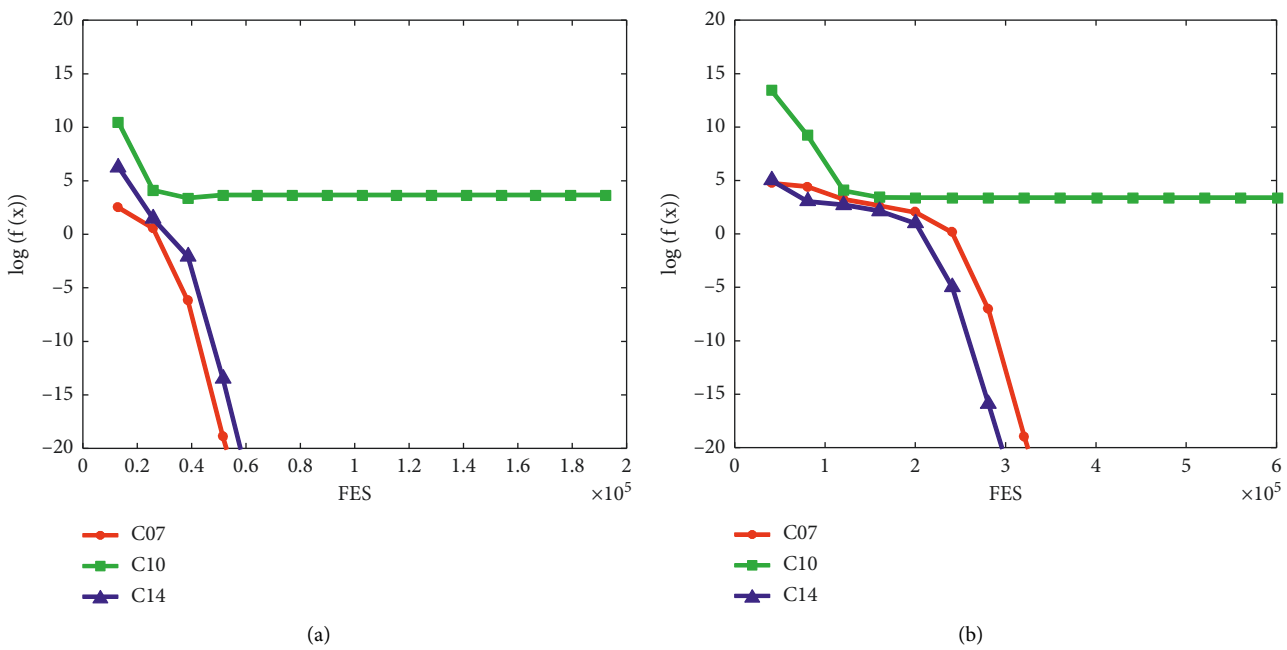


Figure 3: Convergence graphs of HyCO on six representative COPs from IEEE CE C2010. (a) C07, C10, and C14 with 10 d. (b) C07, C10, and C14 with 30 d.

Furthermore, according to the Friedman's test, HyCO obtains the First rank. Based on these considerations, HyCO is superior to other compared methods on 18 COPs with 10 d from CEC 2010.

In terms of $d = 30$, all results are recorded in Tables 5–7, respectively. As described in Table 5, HyCO is superior to FROFI, ITLBO, DeCODE, AIS-IRP, and ECHTDE on 6, 14, 6, 15, and 15 test problems, respectively. In contrast, FROFI, ITLBO, DeCODE, AIS-IRP, and ECHTDE exhibit better performance than HyCO on 1, 0, 2, 2, and 1 test problems, respectively. From Table 6, HyCO obtains the first rank. In addition, according to the multiple-problem Wilcoxon's test, the R^- values cannot exceed the R^+ values in all comparisons, and $\rho \leq 0.05$ can be seen in three cases (i.e., HyCO vs. AIS-IRP, HyCO vs. ECHTDE, and HyCO vs. ITLBO). In summary, HyCO exhibits better performances than other compared methods on 18 COPs with 30 d from CEC 2010.

4.3. Experiments on the 56 COPs from IEEE CEC 2017. To evaluate the performance of HyCO on complicated COPs, 56 high-dimensional COPs from IEEE CEC 2017 are employed. Two methods, which are derived from the competition at IEEE CEC 2017, are selected as the competitors: LSHADE44 [45] and UDE [46]. The results are reported in Table 8. The test functions C17, C18, C19, C26, C27, and C28 cannot find feasible solutions by these three algorithms, and thus they are removed from the comparison.

As described in Table 8, with respect to 28 COPs with 50 d from CEC 2017, HyCO surpasses LSHADE44 and UDE on 12 and 16 test problems, respectively. However, LSHADE44 and UDE provide better results on 6 and 2 test functions, respectively. In terms of 28 COPs with 100 d from CEC 2017, HyCO outperforms LSHADE44 and UDE on 12 and 17 test functions, respectively, while LSHADE44 and UDE perform better than HyCO on 7 and 3 test problems, respectively. Therefore, HyCO exhibits better performance for high-dimensional COPs.

4.4. Visualization of the Evolution Process. The convergence graphs of HyCO on six representative COPs are plotted in Figure 3. As shown in Figure 3, at the early evolving stage, the convergence speed is slow, and the local search model plays an important role in guiding the population to explore more promising areas. Along with the evolution, the convergence rate becomes faster, and some individuals in the population are feasible. At this time, the global search model plays an important role in guiding the population toward the feasible region sufficiently. Therefore, the local and global search models proposed in this paper can achieve a balance between convergence and diversity.

4.5. Sensitivity of Parameter K. The effect of the number of subpopulations K on HyCO is investigated in this section; the numerical experiments are conducted on five different K values: 8, 10, 12, 14, and 16. The results on 18 COPs with 30 d from CEC 2010 are given in Table 9. As shown in Table 9, HyCO achieves the best results when $K = 15$. Specially, HyCO with $K = 15$ provides better results than $K = 8$, $K = 10$, $K = 12$, $K = 14$, and $K = 16$, on 8, 6, 4, 4, and 3 test problems, respectively, While HyCO with $K = 8$, $K = 10$, $K = 12$, $K = 14$, and $K = 16$ perform better than that with $K = 15$ on 0, 1, 0, 0, and 1 test problems, respectively. Therefore, $K = 15$ is a suitable parameter for 18 COPs with 30 d from CEC 2010.

Table 9: Results with different K on 18 COPs with 30 d from CEC 2010.

Problem	Criteria	$K=8$	$K=10$	$K=12$	$K=14$	$K=16$	$K=15$
C01	Mean	−8.17E − 01	−8.20E − 01	−8.19E − 01	−8.20E − 01	−8.19E − 01	−8.19E − 01
	$O \pm S$	± 5.81E − 03−	± 2.48E − 03≈	± 2.58E − 03≈	± 3.25E − 04≈	± 2.34E − 03≈	± 2.83E − 03
C02	Mean	−2.24 E + 00	−2.25 E+ 00	−2.27 E + 00	−2.26 E + 00	−2.26 E + 00	−2.27 E + 00
	$O \pm S$	± 2.61E − 02−	± 2.71E − 02−	± 2.65E − 02≈	± 2.59E − 02−	± 2.47E − 02≈	± 1.04E − 02
C03	Mean	2.73 E + 01	2.64 E + 01	3.00 E + 01	2.52 E + 01	2.86 E + 01	2.52 E + 01
	$O \pm S$	± 1.13 E + 01−	± 7.94 E + 00−	± 2.64 E + 01−	± 9.51 E + 00≈	± 1.42 E + 01−	± 9.51 E + 00
C04	Mean	−3.33E − 06	−3.33E − 06	−3.33E − 06	−3.33E − 06	−3.33E − 06	−3.33E − 06
	$O \pm S$	± 4.66E − 10≈	± 3.92E − 10≈	± 3.00E − 10≈	± 1.28E − 09≈	± 2.45E − 09≈	± 1.84E − 09
C05	Mean	−4.84 E + 02	−4.84 E + 02	−4.84 E+02	−4.84 E + 02	−4.84 E + 02	−4.84 E + 02
	$O \pm S$	± 8.54E − 02≈	± 1.37E − 01≈	± 5.84E − 02≈	± 5.40E − 02≈	± 7.35E − 02≈	± 6.11E − 02
C06	Mean	−5.30 E + 02	−5.30 E + 02	−5.30 E + 02	−5.30 E + 02	−5.30 E + 02	−5.30 E + 02
	$O \pm S$	± 2.88E − 01≈	± 2.10E − 01≈	± 2.52E − 01≈	± 2.70E − 01≈	± 2.53E − 01≈	± 1.52E − 01
C07	Mean	1.59E − 01	2.21E − 27	3.19E − 01	1.59E − 01	1.64E − 28	1.91E − 27
	$O \pm S$	± 7.97E − 01−	± 5.17E − 27≈	± 1.10 E + 00−	± 1.32E − 01−	± 3.45E − 28≈	± 7.27E − 27
C08	Mean	5.43E − 28	1.11E − 27	2.74E − 27	1.59E − 01	3.42E − 27	1.65E − 27
	$O \pm S$	± 2.72E − 27≈	± 3.83E − 27≈	± 5.58E − 27≈	± 7.97E − 01−	± 1.48E − 27≈	± 4.56E − 27
C09	Mean	2.39 E + 01	1.75 E + 01	1.24 E + 01	3.64 E + 00	3.26 E + 00	1.76E − 01
	$O \pm S$	± 3.39 E + 01−	± 3.07 E + 01−	± 2.74 E + 01−	± 1.41 E + 01−	± 1.36 E + 01−	± 8.80E − 01
C10	Mean	3.13 E + 01	3.13 E + 01	3.13 E + 01	3.13 E + 01	3.13 E + 01	3.13 E + 01
	$O \pm S$	± 1.17E − 05≈	± 3.15E − 04≈	± 4.23E − 06≈	± 4.14E − 06≈	± 3.72E − 06≈	± 1.68E − 06
C11	Mean	−3.92E − 04	−3.83E − 04	−3.92E − 04	−3.92E − 04	−3.92E − 04	−3.92E − 04
	$O \pm S$	± 2.07E − 10∇−	± 4.53E − 05−	± 2.46E − 10≈	± 2.93E − 10≈	± 3.65E − 09≈	± 6.11E − 10
C12	Mean	−1.99E − 01	−1.99E − 01	−1.99E − 01	−1.99E − 01	−1.99E − 01	−1.99E − 01
	$O \pm S$	± 1.24E − 08≈	± 7.54E − 07≈	± 3.68E − 03≈	± 3.52E − 04≈	± 2.31E − 09≈	± 2.66E − 08
C13	Mean	−6.33 E + 01	−6.26 E + 01	−6.25 E + 01	−6.35 E + 01	−6.26 E + 01	−6.36 E + 01
	$O \pm S$	± 2.33 E + 00≈	± 2.40 E + 00−	± 2.54 E + 00−	± 2.15E − 00≈	± 2.09 E + 00−	± 1.93 E + 00
C14	Mean	5.57E − 29	6.38E − 01	0.00 E + 00	0.00 E + 00	1.36E − 28	1.29E − 27
	$O \pm S$	± 2.78E − 28≈	± 1.49 E + 00−	± 0.00 E + 00≈	± 0.00 E + 00≈	± 2.01E − 28≈	± 3.88E − 27
C15	Mean	2.21 E + 01	2.16 E + 01	2.16 E + 01	2.16 E + 01	2.16 E + 01	2.16 E + 01
	$O \pm S$	± 1.58 E + 00−	± 1.33E − 07≈	± 3.66E − 07≈	± 3.16E − 07≈	± 1.69E − 07≈	± 1.87E − 07
C16	Mean	0.00 E + 00	0.00 E + 00	0.00 E + 00	0.00 E + 00	0.00 E + 00	0.00 E + 00
	$O \pm S$	± 0.00 E + 00≈	± 0.00 E + 00≈	± 0.00 E + 00≈	± 0.00 E + 00≈	± 0.00 E + 00≈	± 0.00 E + 00
C17	Mean	9.65E − 01	3.36E − 30	2.70E − 06	3.36E − 11	9.91E − 21	4.97E − 12
	$O \pm S$	± 1.73 E + 00−	± 1.68E − 30+	± 1.35E − 05≈	± 1.68E − 10≈	± 4.35E − 20+	± 1.87E − 11
C18	Mean	1.05E − 30	1.19E − 31	5.88E − 32	1.16E − 31	4.39E − 30	7.14E − 32
	$O \pm S$	± 4.75E − 30≈	± 1.19E − 31≈	± 6.31E − 32≈	± 1.23E − 31≈	± 4.65E − 30≈	± 1.02E − 31
−		8	6	4	4	3	
+		0	1	0	0	1	
≈		10	11	14	14	14	

Table 10: Experimental results of HyCO and CMODE for five engineering design problems.

Problem	Criteria	CMODE	HyCO
Three-bar truss design problem	Mean $O \pm S$	2.66 E + 02 ± 1.44 E + 00−	2.65 E + 02 ± 8.90E − 01
Pressure vessel design problem	Mean $O \pm S$	5.89 E + 03 ± 1.69E − 04≈	5.89 E + 03 ± 1.38 E + 00
Tension/compression spring design problem	Mean $O \pm S$	7.17E − 03 ± 8.52E − 06−	7.15E − 03 ± 1.57E − 06
Speed reducer design problem	Mean $O \pm S$	3.01 E + 03 ± 3.63 E + 00−	3.00 E + 03 ± 1.48 E + 00
Gear train design problem	Mean $O \pm S$	3.40E − 02 ± 1.89E − 05≈	3.40E − 02 ± 7.08E − 18
−		3	
+		0	
≈		2	

4.6. Real-World Application. To test the performance of HyCO in real-world COPs, five engineering design problems are adopted. The details of these five engineering problems are obtained from the literature [47]. CMODE [48], which is a representative constrained optimization method, is selected as a competitor. The maximum number of evaluations of these five problems are set to 500, 70000, 10000, 10000, and 5000, respectively. The population size and the number of subpopulations are set to 100 and 15. The parameters of CMODE are consistent with the original literature. The results of these two methods are reported in Table 10.

As shown in Table 10, HyCO outperforms CMODE on 3 engineering design problems, while CMODE cannot be better than HyCO in any problems. In summary, HyCO is effective for solving the real-world engineering optimization problems.

5. Conclusion

In this paper, HyCO is designed to solve COPs. In the method, the local and global search models are designed to balance both diversity and convergence. To balance constraints and objective function, the direction vector is adjusted according to the direction vector adjustment strategy. Experiment results on three benchmark test suites, namely, 36 COPs from IEEE CEC 2010, 56 COPs from IEEE CEC 2017, and 5 real world engineering design issues, demonstrate the following conclusions: (1) HyCO is competitive than other selected methods. (2) The local and global search models can achieve the balance between diversity and convergence. (3) The direction vector adjustment strategy can guide the population to converge to the feasible optimal solution.

In our future research, it is meaningful to design a self-adaptive the direction vector adjustment strategy in HyCO to solve high-dimensional test functions. In addition, online learning [49–51] will be introduced into constraint optimization in the future.

Data Availability

Data and code are available upon request through sending an e-mail to gaoweifeng2004@126.com.

Conflicts of Interest

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

Acknowledgments

This work was supported in part by the National Nature Science Foundation of China under Grants 61772391 and 62106186, in part by the Natural Science Basic Research Plan in Shaanxi Province of China under Grant 2022JQ-670, and in part by the Fundamental Research Funds for the Central Universities under Grant QTZX22047.

References

[1] L. Ma, X. Wang, M. Huang, Z. Lin, L. Tian, and H. Chen, "Two-level master slave rfid networks planning via hybrid multi-objective artificial bee colony optimizer," *IEEE Transactions on Systems*, vol. 49, no. 5, pp. 1–20, 2017.

[2] Y. H. Jia, W. N. Chen, T. Gu et al., "A dynamic logistic dispatching system with set-based particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 9, pp. 1607–1621, 2018.

[3] L. Jinjian, A. Doniec, and J. Boonaert, "A New Integrated Multimodal Multi-Criteria Route Planning 348 Method in Bus Network Considering Passengers's Walking Behavior," in *Proceeding of the 2018 3rd IEEE International Conference 349 on Intelligent Transportation Engineering (ICITE)*, pp. 13–17.

[4] B. Yen and G. G. Yen, "An adaptive penalty formulation for constrained evolutionary optimization," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 39, no. 3, pp. 565–578, 2009.

[5] C. Yu, K. Lay Teo, L. Zhang, and Y. Bai, "A new exact penalty function method for continuous inequality constrained optimization problems," *Journal of Industrial and Management Optimization*, vol. 6, no. 4, pp. 895–910, 2010.

[6] O. Yeniay, "Penalty function methods for constrained optimization with genetic algorithms," *Mathematical and Computational Applications*, vol. 10, no. 1, pp. 45–56, 2005.

[7] M. M. Ali and W. X. Zhu, "A penalty function-based differential evolution algorithm for constrained global optimization," *Computational Optimization and Applications*, vol. 54, no. 3, pp. 707–739, 2013.

[8] C. A. Coello Coello, "Constraint-handling using an evolutionary multiobjective optimization technique," *Civil Engineering and Environmental Systems*, vol. 17, no. 4, pp. 319–346, 2000.

[9] T. Liew and K. M. Liew, "Society and civilization: an optimization algorithm based on the simulation of social behavior," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 4, pp. 386–396, 2003.

[10] Z. Wang and Y. Wang, "A multiobjective optimization-based evolutionary algorithm for constrained optimization," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 658–675, 2006.

[11] Y. Wang, Z. Cai, G. Zhou, and Y. Zhou, "Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 3, pp. 560–575, 2007.

[12] C. A. C. Coello and E. M. Montes, "Constraint-handling in genetic algorithms through the use of ominance-based tournament selection," *Advanced Engineering Informatics*, vol. 16, no. 3, pp. 193–203, 2002.

[13] L. Ma, M. Huang, S. Yang, R. Wang, and X. Wang, "An adaptive localized decision variable analysis approach to large-scale multiobjective and many-objective optimization," *IEEE Transactions on Cybernetics*, vol. 52, no. 7, pp. 6684–6696, 2022.

[14] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2-4, pp. 311–338, 2000.

[15] T. Sakai and S. Sakai, "Constrained optimization by applying the<tex>$alpha$</tex>Constrained method to the nonlinear simplex method with mutations," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 5, pp. 437–451, 2005.

[16] T. P. Xin Yao and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284–294, 2000.

[17] K. Rahi, H. Singh, and T. Ray, "Partial evaluation strategies for expensive evolutionary constrained optimization," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 6, pp. 1103–1117, 2021.

[18] G. Li and Q. Zhang, "Multiple penalties and multiple local surrogates for expensive constrained optimization," *IEEE Transactions on Neural Networks/A Publication of The IEEE Neural Networks Council*, vol. 25, no. 4, pp. 769–778, 2021.

[19] G. Li, Q. Lin, Q. Gao, and W. Gao, "A three-level radial basis function method for expensive optimization," *IEEE Transactions on Cybernetics*, vol. 52, no. 7, pp. 5720–5731, 2022.

[20] T. Takahama and S. Sakai, "Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites," in *Proceeding of the IEEE International Conference on Evolutionary Computation*, pp. 1–8, Vancouver, BC, Canada, July 2006.

[21] J. Brest, B. Boskovic, and V. Zumer, "An Improved self-adaptive differential evolution algorithm in single objective constrained real-Parameter optimization," in *Proceeding of the IEEE Congress on Evolutionary Computation*, pp. 1–8, Barcelona, Spain, July 2010.

[22] T. Takahama and S. Sakai, "Efficient constrained optimization by the ε constrained adaptive differential evolution," in *Proceeding of the IEEE Congress on Evolutionary Computation*, pp. 1–8, Barcelona, Spain, July 2010.

[23] C. Peng, H. L. Gu, and F. Gu, "A novel constraint-handling technique based on dynamic weights for constrained optimization problems," *Soft Computing*, vol. 22, no. 12, pp. 3919–3935, 2018.

[24] A. Mani and C. Patvardhan, "A novel hybrid constraint handling technique for evolutionary optimization," in *Proceeding of the IEEE Congress on Evolutionary Computation*, pp. 2577–2583, Trondheim, Norway, May 2009.

[25] Y. Yong Wang, Z. Zixing Cai, Y. Yuren Zhou, and Wei Zeng, "An adaptive tradeoff model for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 80–92, 2008.

[26] G. Li, Q. Zhang, and Z. Wang, "Evolutionary competitive multitasking optimization," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 2, pp. 278–289, 2022.

[27] Y. Zixing Cai and Z. Cai, "A dynamic hybrid framework for constrained evolutionary optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 1, pp. 203–217, 2012.

[28] W. F. Gao, G. G. Liu, and S. Y. Liu, "A dual-population differential evolution with coevolution for constrained optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 5, pp. 1108–1121, 2015.

[29] Y. Wang, J. P. Li, X. Xue, and B. C. Wang, "Utilizing the correlation between constraints and objective function for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 29–43, 2019.

[30] B. C. Wang, H. X. Li, and Q. Zhang, "Decomposition-based multiobjective optimization for constrained evolutionary optimization," *IEEE Transactions on Systems*, pp. 1–14, 2018.

[31] B. C. Wang, H. X. Li, J. P. Li, and Y. Wang, "Composite differential evolution for constrained evolutionary optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 7, pp. 1482–1495, 2019.

[32] X. Cai, Z. Yang, Z. Zhang, and Q. Zhang, "Decomposition-based-sorting and angle-based-selection for evolutionary multiobjective and many-objective optimization," *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2824–2837, 2017.

[33] R. Cheng, Y. Jin, M. Sendhoff, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 773–791, 2016.

[34] S. Suganthan and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[35] Y. Wang, B. C. Wang, H. X. Li, and G. G. Yen, "Incorporating objective function information into the feasibility rule for constrained evolutionary optimization," *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 2938–2952, 2016.

[36] B. Wang, H. Li, J. Li, and J. Wang, "Composite differential evolution for constrained evolutionary optimization," *IEEE Transactions on Systems Man & Cybernetics Systems*, vol. 49, no. 7, pp. 1482–1495, 2019.

[37] Y. Wang, J. P. Li, and X. Wang, "Utilizing the correlation between constraints and objective function for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 29–43, 2020.

[38] Q. Hui Li and H. Li, "Moea/d: a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[39] R. Mallipeddi and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization," *Nanyang Technological University*, 2010.

[40] Y. Wang, B. C. Wang, H. X. Yen, and G. G. Yen, "Incorporating objective function information into the feasibility rule for constrained evolutionary optimization," *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 2938–2952, 2016.

[41] B. Wang, H. X. Feng, and Y. Feng, "An improved teaching-learning-based optimization for constrained evolutionary optimization," *Information Sciences*, vol. 456, pp. 131–144, 2018.

[42] W. Zhang, G. G. He, and Z. He, "Constrained optimization via artificial immune systemficial immune system," *IEEE Transactions on Cybernetics*, vol. 44, no. 2, pp. 185–198, 2014.

[43] R. Mallipeddi and P. N. Suganthan, "Differential evolution with ensemble of constraint handling techniques for solving

CEC 2010 benchmark problems," in *Proceeding of the IEEE Congress on Evolutionary Computation*, pp. 1–8, Barcelona, Spain, July 2010.

[44] J. Alcalá-Fdez, L. Sánchez, S. del Jesus, J. C. VenturaGarrellOteroRomeroBacarditRivasFerná ndez-Herrera, and F. Herrera, "Keel: a software tool to assess evolutionary algorithms for data mining problems," *Soft Computing*, vol. 13, no. 3, pp. 307–318, 2009.

[45] R. Polakova, "L-Shade with competing strategies applied to constrained optimization," in *Proceeding of the IEEE Congress on Evolutionary Computation*, pp. 1683–1689, Donostia, Spain, June 2017.

[46] A. Trivedi, K. Sanyal, P. Verma, and D. Srinivasan, "A unified differential evolution algorithm for constrained optimization problems," in *Proceeding of the IEEE Congress on Evolutionary Computation*, pp. 1231–1238, Donostia, Spain, June 2017.

[47] A. Sadollah, A. Bahreininejad, H. Hamdi, and M. Hamdi, "Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems," *Applied Soft Computing*, vol. 13, no. 5, pp. 2592–2612, 2013.

[48] Y. Cai and Z. Cai, "Combining multiobjective optimization with differential evolution to solve constrained optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, pp. 117–134, 2012.

[49] L. Ma, L. Nan, Y. Guo et al., "Learning to optimize: reference vector reinforcement learning adaption to constrained many-objective optimization of industrial copper burdening system," *IEEE Transactions on Cybernetics*, pp. 1–14, 2021.

[50] L. Ma, S. Cheng, and Y. Shi, "Enhancing learning efficiency of brain storm optimization via orthogonal learning design," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 11, pp. 6723–6742, 2021.

[51] X. He, C. Fei, Y. Liu, K. Yang, and X. Ji, "Multi-objective longitudinal decision-making for autonomous electric vehicle: a 337 entropy-constrained reinforcement learning approach," in *Proceeding of the IEEE 23rd International Conference on Intelligent 338 Transportation Systems (ITSC)*, pp. 1–6, September 2020.

Hindawi
Discrete Dynamics in Nature and Society
Volume 2022, Article ID 8545739, 10 pages
https://doi.org/10.1155/2022/8545739

Research Article

Parameters Optimization of Multipass Milling Process by an Effective Modified Particle Swarm Optimization Algorithm

Cuiyu Wang, Wenwen Wang, Yiping Gao, and Xinyu Li

School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, China

Correspondence should be addressed to Yiping Gao; gaoyiping@hust.edu.cn

Received 28 June 2022; Revised 31 July 2022; Accepted 8 August 2022; Published 19 September 2022

Academic Editor: Xueyi Wang

The selection of the optimal metal milling parameters greatly impacts final product quality and production efficiency in modern manufacturing systems. The profit rate is also sensitive to the selected parameters. This research focuses on determining the optimal parameters of a multipass milling process using an improved particle swarm optimization (PSO) method. The objective is to minimize the production time. The proper number of passes, the optimal cut speed, and feed rate are considered as the parameters (the decision variables in the model) which are needed to be optimized. Furthermore, the permissive arbor strength, arbor deflection, and motor power are the constraints of the model. The penalty function method is used as the constraints handling technique to address the constraints efficiently in the proposed method. A case is adopted and solved to evaluate the performance of the proposed method. The experimental part is analyzed and compared with advanced methods. Experimental results show that the proposed method is very effective for parameters optimization of a multipass milling process and outperforms other methods.

1. Introduction

In recent years, optimal machining parameters for milling process play an important role to ensure the quality of final products, productivity, and the competition of the manufacturers [1]. As a result, determining the optimal milling parameters becomes a very hot research topic. Metal milling is one of the most important and widely used manufacturing processes. But it is difficult to choose the optimal milling parameters which are subject to multiconstraints of motor power, cutting force, and feasible range of machining parameters [2]. The single-pass milling process was originally preferred to find the minimum production time for economic reasons. However, with the development of the economy, the single-pass process cannot meet the needs of product quality [3]. Hence more and more researchers tend to focus on the multipass machining operation. Traditionally, the operators usually depend on the handbooks, and if the machines and processing types do not exist, the operators can only rely on their experiences to choose the parameters. However, handbooks and experiences cannot guarantee the selected parameters are optimal and this may lead to the waste of time and resources [4].

Several researchers used mathematical methods to find the authentic optimal machining parameters. The traditional strategies such as dynamic programming and geometric programming (GP) have been applied to the optimization of the single-pass cutting process [5]. Tolouei-Rad and Bidhendi [6] used a feasible direction method to determine the optimal parameters in milling operation. Petropoulos [7] used a GP method for practical optimization problems. Although the application of traditional methods mentioned above obtained better results than those gained by just following the handbooks, the results were far from robust when compared with the results obtained by heuristic approaches.

However, these traditional methods cannot be the optimal strategies to determine the optimal machining parameters, especially for the multipass process. This is because milling parameters optimization is a complicated multiconstraints nonlinear programming problem. Traditional methods rely on gradient information and are far from

robust. So many researchers begin to apply heuristic methods to this problem. Vijayakumar et al. [8] used ant colony optimization (ACO) to find the satisfactory parameters in multipass turning operation. Palanisamy et al. [9] and Gakhe et al. [10] used a genetic algorithm (GA) to obtain the optimal machining parameters and minimize the total production time. Wang et al. [11] hybridized GA and SA and got the new strategy GSA, to select the optimal parameters in milling operation. The artificial bee colony (ABC) [12] was applied to the milling parameters optimization problem by Rao et al. [5]. Baykasoglu [13] proposed a weighted superposition attraction-repulsion algorithm for cutting conditions.

Particle swarm optimization (PSO) is a soft computation technology based on swarm intelligence [14, 15]. It can be used in the area of system design, multiobjective optimization, pattern recognition, signal processing, games, decision making, simulation, and identification [16, 17]. PSO had also been used to identify human tremors by Eberhart et al. [18], and the computational process was fast and the result was accurate. Han et al. [19] introduced PSO into the flexible manufacturing system to find the optimal scheduling and increase the use ratio of machines and productivity. Tang et al. [20] developed a parallel random matrix PSO for scheduling with budget constraints. Yang et al. [21] proposed an improved multiobjective PSO for scheduling with panel block construction. Sun and Ren [22] introduced graph density into PSO. PSO also had been used in the fields of CNC machining [23] and pulsed laser micromachining [24], and both gained good results. Fang et al. [25] proposed an improved adaptive PSO for cutting parameter optimization.

PSO is simple in concept, convenient in computation, fast in implementation, and brief in programming [26]. Using PSO to figure out nonlinear programming problems with multiple constraints is efficient. That is why PSO is considered to be one of the most popular methods among nature-inspired methods to solve complex optimization problems. However, the balance of the exploration (diversification) and exploitation (intensification) is one of the limitations, which impedes the application of PSO in the multipass milling process greatly. Hence, an effective modified particle swarm optimization method is proposed for the multipass milling process. The motivation of the proposed method is to balance the exploration (diversification) and exploitation (intensification) of PSO, which aims to improve the optimization results. With this motivation, three aspects of improvement, including the searchability, avoiding the local optima, and restart parameter, have been implemented, and the constraints handling strategy is introduced. Moreover, the penalty function method is combined with PSO to optimize the parameters optimization of the multipass milling process. The main contribution of this paper has several points. Firstly, in order to improve the PSO and avoid the local optima, the proposed method introduces three aspects of modification to balance the diversification and intensification. Secondly, a constraints handling strategy is introduced to construct the penalty function for the feasible particle. Finally, the proposed method is developed into a multipass milling process optimization problem, which has indicated the effectiveness of the proposed method.

The rest of this paper is organized as follows: the problem is formulated in Section 2. Section 3 introduces the proposed improved PSO. In Section 4, an application example has been selected to testify to the performance of the proposed PSO and the results are compared with the ones gained by other strategies. Some conclusions are given in the last section.

2. Problem Formulation

This paper uses the mathematical model proposed by [27]. The milling parameters optimization model is made up of three essential parts, which are objective function, constraints condition, and decision variables. In this model, the total production time is the objective function. The number of passes, the depth of cut for the rough pass and final pass, cutting speed, and feed per teeth are the parameters to be optimized. The constraints of this problem include the arbor strength, arbor deflection, and permissible motor power.

2.1. Objective Function. In this multipass milling model, a component's milling process contains some rough processes and one finishing process. The rough processes fulfil the removal of a large number of materials while the finishing process ensures the permissible surface roughness [5].

The total production time for a component is T_{pr}; it is composed of the following items:

$$T_{pr} = T_p + T_L + T_\alpha + T_m + T_c. \tag{1}$$

T_{p} is the machine set-up time. $T_p = T_s/N_b$, where T_s is the set-up time for a new batch and N_b means the total number of components in a produced batch.

T_L is the loading-unloading time for one part.

T_α is the process of adjusting and quick return time.

T_{m} is the actual machining time. $T_m = L/f$, where L (mm) is the cutting length and f (mm/min) means the feed rate. $f = f_z \cdot Z \cdot N$, where f_z stands for the feed per tooth (mm/tooth), Z refers to the total number of teeth on the cutter, N means spindle speed (rpm), and N can be given as $N = 1000 \cdot \mathrm{V}/\pi \mathrm{D}$, where V is cutting speed (m/min) and D means cut diameter (mm).

T_{c} is the time per component to change machining tool. $T_c = T_d T_m/T$, where T_d means actual time for changing a dull cutting edge, and $T = (C_V^{1/m} D^{b_v/m}/V^{1/m} \alpha^{e_v/m} f_z^{u_v/m} \alpha_w^{r_v/m} Z^{n_v/m} \lambda_S^{q_v/m})(\mathrm{B_m B_h B_p B_t})^{1/m}$ represent tool life. For a multipass milling process, the total machining time should take every milling pass into consideration that T_{pr} becomes

$$T_{pr} = \frac{T_S}{N_b} + T_L + N_P T_a + \sum_{i=1}^{N_p}\left(\frac{\pi DL}{1000 f_{zi} Z V_i} + \frac{T_d \pi L V_i^{1/m-1} a_i^{e_v/m} f_{zi}^{u_v/m-1} a_w^{r_v/m} Z^{n_v/m-1} \lambda_s^{q_v/m}}{1000 C_V^{1/m} D^{b_v/m-1}\left(B_m B_h B_p B_t\right)^{1/m}}\right). \quad (2)$$

N_p is the number of passes to remove the total depth of cut.

i is a subscript that refers to the i-th pass of a process.

$T_1 = T_s/N_b + T_L$ and $T_2 = \sum(T_{\alpha i} + T_{mi} + T_d T_{mi}/T)$.

2.2. Constraint Conditions. Feasible results are subject to some technological constraints that are related to the machining tools, cutting tools, and materials. Design variables are bounded by the following constraints [27].

2.2.1. Power Constraint. Tool power of machine ought to be less than effective power, which is shown as

$$P_c \le P_m \eta. \quad (3)$$

And for plain milling case,

$$C_{zp} \cdot \alpha_w \cdot Z \cdot D^{b_z} \alpha^{e_z} f_z^{u_z} V/6120 \le P_m \eta. \quad (4)$$

P_c is the cutting power

P_m is the nominal motor power

η is the overall efficiency of the cutting machine

F_c is the cutting force which can be calculated according to $C_{zp} \cdot \alpha_w \cdot Z \cdot D^{b_z} \alpha^{e_z} f_z^{u_z}$

2.2.2. Arbor Strength. Cutting force F_c should not exceed the permissible strength F_s, and all the coefficients are following [27]:

$$F_c - F_s \le 0,$$

$$F_c - \left(\frac{0.1 k_b d_\alpha^3}{0.08 L_\alpha + 0.65\sqrt{\left(0.25 L_\alpha\right)^2 + \left(0.5 D k_b/1.3 k_t\right)}}\right) \le 0. \quad (5)$$

F_c is the cutting force

F_s is the permissible force for the arbor strength

k_b is the permissible bending strength of the arbor

d_α is the diameter of the arbor

L_α is the arbor length between supports

k_t is the permissible torsional stress of the arbor material

2.2.3. Arbor Deflection. Arbor deflection must be checked when choosing the satisfying feed rate:

$$F_c - F_d \le 0,$$

$$F_d = 4\frac{Eed_a^4}{L_a^3}. \quad (6)$$

F_d is the permissible strength of the arbor deflection

E is the modulus of elasticity of the arbor material

e is the permissible value of the arbor deflection, and $e = 0.2$ in roughing pass meanwhile $e = 0.2$ in finishing pass

2.3. Decision Variables. The number of passes, the cut depth of each pass, the feed rate, and the feed per tooth are design variables. The last two parameters can be gained from the range of spindle speed and spindle feed rate. The first two parameters are determined by the permissible boundary of cutting depth and the total milling depth. The feasible range of depth of cut is specified as [5]

$$\alpha_{\min} \le \alpha \le \alpha_{\max}, \quad (7)$$

where $\alpha_{\min}$ and $\alpha_{\max}$ determine the boundary of the choice of the depth of cut in every pass.

Yet the maximum spindle speed $N_{\max}$ and the minimum spindle speed $N_{\min}$ determine the range of milling speed:

$$V_{\min} \le V \le V_{\max}, \quad (8)$$

where $V_{\min} = \pi D N_{\min}/1000$ and $V_{\max} = \pi D N_{\max}/1000$.

Also, the feasible feed rate must be in the range determined by the maximum and minimum feed rate of the machine.

$$f_{z\min} \le f_z \le f_{z\max}, \quad (9)$$

where $f_{z\min} = f_{\min}/ZN_{\max}$ and $f_{z\max} = f_{\max}/ZN_{\min}$ and $f_{\max}$ are the minimum and maximum spindle feed rate. The PSO will be introduced in the following section based on the given multipass milling model.

3. Proposed PSO for Parameter Optimization

3.1. Conventional PSO for Parameter Optimization. PSO is a swarm-intelligence-based algorithm and was first proposed as the simulation of a bird flock's social behavior. PSO randomly generates a set of birds with random velocities and random positions. After that, an initial group of swarms is produced. Individuals in the population are particles. The number of particles in the swarm is the population size. Every particle searches for the best position to get the minimum value of the objective function just like the bird searching for food. The search space of the optimization problem is bounded by the feasible range of each decision

variable. The position of each particle represents a potential solution because every dimension of the search space stands for the corresponding design variable. Each particle keeps track of its coordinates in the problem space which is associated with the best solution it has achieved so far [19]; it is p_{best}. Another important value that is tracked is the overall best value. The overall best location is called g_{best}. Fitness function is the criterion to judge the positions of particles. And fitness function is composed of the objective function and the constraints. p_{best} and g_{best} are continually updated by the better locations and better fitness in the searching process. This searching process is exactly like the self-learning and social-learning behavior of birds. In a word, the PSO concept consists of each time stop. Each particle changing velocity toward P_{best} and g_{best} randomly generates the acceleration for each best position. Every particle modifies its position and velocity after one search time by the equations given as follows:

$$v_i^{t+1} = \omega \cdot v_i^t + c_1 r_1 \left(P_{\text{best}} - P_i^t\right) + c_2 r_2 \left(G_{\text{best}_i} - p_i^t\right) \quad i = 1, \ldots, d, \tag{10}$$

$$p_i^{t+1} = p_i^t + v_i^{t+1}, \quad i = 1, \ldots, d. \tag{11}$$

d is the dimension of the optimization problem (the number of design variables)

$v_i = (v_1, v_2, \ldots, v_d)$ is the velocity of particles in the d-dimension search space

$p_i = (p_1, p_2, \ldots, p_d)$ is the i-th particle's d-dimension position

$p_{\text{best}_i} = (p_{\text{best}_1}, p_{\text{best}_2}, \ldots, p_{\text{best}_d})$ is the personal best position

$g_{\text{best}_i} = (g_{\text{best}_1}, g_{\text{best}_2}, \ldots, g_{\text{best}_d})$ is the global best position

ω means the inertia weight controls the particle's exploration and exploitation ability

c_1, c_2 are the acceleration constants that represent the weighting that pulls each particle toward p_{best} and g_{best} position

r_1, r_2 are two uniform random numbers between [0, 1]

The process for implementing PSO is as follows:

(i) Initialize the population of particles with random positions and random velocities within the search ranges

(ii) Evaluate each particle with both the objective function and constraints

(iii) Compare each particle's objective value with its p_{best}, take their constraints satisfaction into consideration, and if the current position is better, take the place of the previous fitness value and p_{best}

(iv) Compare each particle fitness evaluation and constraints satisfaction state with the overall previous best g_{best}, updating g_{best} if it exits better particle and keeping in track of the best fitness value

(v) Change the velocity and position of each particle by (10) and (11)

(vi) Loop to (ii) if the maximum number of iterations is not reached or the given precision is met

3.2. Constraints Handling Technique. In PSO, a traditional method to deal with the constraints is usually adopted by Yang et al. [28]. In this method, the more constraints a particle violates, the worse the particle is. The degree of violation of constraints is neglected. So, it is not so easy for this method to judge the particles. The strategy is shown in Table 1.

The constraints handling technique is operated in the encoding work rather than be put into the objective function. But it becomes hard to ensure the results are feasible when the number of constraints increases. The final optimal particle may just violate fewer constraints but is still not feasible.

Thus, the penalty function method is adopted in this study. The overall constraints are contained in the penalty function. The fitness function is composed of the penalty function and the objective function. The results are usually feasible and subject to all the constraints when setting the punishment factor to a big enough number. The details to build the fitness function with the penalty function are introduced next.

For a general optimization problem, the mathematical expressions can be described as follows [29].

Assume that $x = (x_1, x_2, \ldots, x_n)$ is a point in n-dimension space R^n and $f(x)$, $h_{\text{i}}(\text{x})$, and $g_{\text{i}}(\text{x})$ are given functions with n variables. Consider the constraints as follows:

$$\begin{aligned} h_i(x)\,(i = 1, \ldots, k) &= 0, \\ g_i(x)\,(i = 1, \ldots, p) &\le 0. \end{aligned} \tag{12}$$

The optimization target is to obtain the minimum $f(\text{x})$:

$f(x)$ is the objective function

$h_i(x)$ is k equality constraints

$g_i(x)$ is p inequality constraints

x are the design variables

The fitness function is given as follows when considering c:

$$F(x) = f(x) + \delta P(x). \tag{13}$$

δ is the punishment factor

$P(x)$ is the punishment term

Without the ε (a positive tolerance value for equality constraints) in [29], the penalty function can be written as

$$P(x) = m_1 \sum_{i=1}^{k} \left|h_i(x)\right|^2 + m_2 \sum_{i=1}^{p} \left\{\max\left(0, g_i(x)\right)\right\}^2. \tag{14}$$

m_1 is the weight of equality constraints

m_2 is the weight of inequality constraints

Table 1: The commonly used updating strategy for p_{best}.

Updating strategy for the personal best position (compare current fitness value and p_{best})
(i) If these two solutions are both feasible, select the one with better fitness value; if both two fitness values are equal, randomly select one of them. (ii) If one is feasible and the other is infeasible, select the feasible one. (iii) If these two solutions are both infeasible, select the one with the lower number of constraints violations; if they violate the same number of constraints, randomly select one of them.

The fitness function is built based on (13) and (14) with the constraints and objective function. The process of building this structure is simple, fast, and time-saving. The results are very stable while using this penalty function method.

3.3. The Proposed PSO. Based on the standard PSO, a new improved PSO is applied to this study. To obtain better computational results and avoid quick convergence, three improvements are applied to standard PSO. A new fitness function based on the problem is constructed.

3.3.1. Modification of PSO

(1) Particles no longer learn from one's own p_{best}. Parts of the swarm obtain p_{best} of other members. The different p_{best} guide the particle flying in another direction. The searchability of the swarm is enhanced. In this study, ten percent of particles were randomly chosen to implement this modification and change p_{best} used in (10).

(2) To avoid the particles getting trapped into local optima, after an iteration of the proposed method, the particles do not update their position and velocity by (12) and (13). Instead, ten percent of particles implement their initialization and randomly get their positions and velocities. In this way, that the majority of the experience has been held back and delivered to the next generation. At the same time, the diversity of the swarm is kept.

(3) The last modification of the standard PSO is a restart parameter to stop its quick convergence. When all the particles go to the same direction guided by g_{best}, the swarm always have a fast convergence rate before finding the optimal point. To get thorough flying in the search space, the particles have some probability to reset and restart the method so the particles can jump out of local optima.

When g_{best} does not update, a counter-repeat times (*RepeatTime*) increase by 1. When the repeat times reach the max repeat time (*MaxRepeatTime*) that means the particles may trap in the local minimum. We set a parameter *rs* that combines with the present iteration times and the max iteration times (*rs* is *present iteration times*/max *iteration times*). Another parameter *rand* is a randomly generated number in $[0, 1]$. On the condition that the max repeat times are reached, compare *rs* with *rand*. If *rs* is smaller than *rand*, all particles have an initialized value and all parameters are reset. But in each restart, the best global fitness is saved. It can be seen that the restart probability decreased with the iteration time rising because *rs* is increased by each iteration.

3.3.2. Fitness Function. In this optimization problem, there are only three inequality constraints. Based on the standard PSO and recommended constraints handling technique, the constraints in (13) and (14) can be substituted with the milling model. The fitness function is shown as follows:

$$\text{Fit} = T_{pr} + m_2\left[\max\left(0, \left(\frac{F_c V}{6120} - P_m \eta\right)\right)^2 + \max\left(0, \left(F_c - F_s\right)\right)^2 + \max\left(0, \left(F_c - F_d\right)\right)^2\right]. \tag{15}$$

According to (2), the fitness function above is a quaternion function (about N_p, α_i, V_i, f_{zi}). So, this is a four-dimension optimization problem. Nevertheless, the optimization results of N_{p} and α_{i} cannot always be implemented in practice, because the removal of each pass (α_{i}) depends on the machining accuracy of the machine. And the number of passes of the cutting process must be an integer. For the reasons above, the two parameters, N_{p} and α_{i}, are optimized by another strategy.

There are some frequently used machining options of N_{p} and α_{i}. α_{i} is defined in the range of $(\alpha_{\min}, \alpha_{\max})$. Then the N_p is fixed by the total cut depth (α) and α_i, $N_{\text{p}} = \alpha/\alpha_{\text{i}}$. Usually, four or five strategies are implemented to obtain the final best results. The set of N_p and α_i that obtain the minimum machining time is considered as the optimal parameters.

The representation of this milling problem by particles is shown in Figure 1.

3.3.3. Computational Process. In this study, the programming language is Microsoft Visual C++ and the flow chart of the proposed PSO is shown in Figure 2.

3.3.4. Complexity of the Proposed Method. Assuming the population size is K, the maximum iteration times is L (*maximum iteration times*), the maximum repetition times is R (*maximum repeat times*), and the problem size is N (N *parameters need to be optimized*), and the computational complexity is analyzed as follows: at each iteration, problem modeling, fitness calculation, optimal state selection of

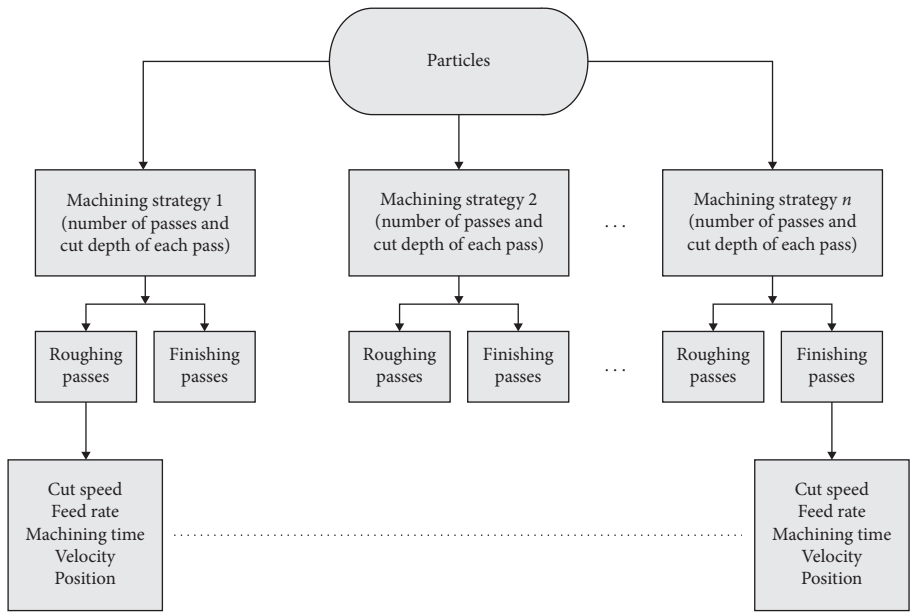


Figure 1: The representation of a solution to the milling parameters optimization problem.

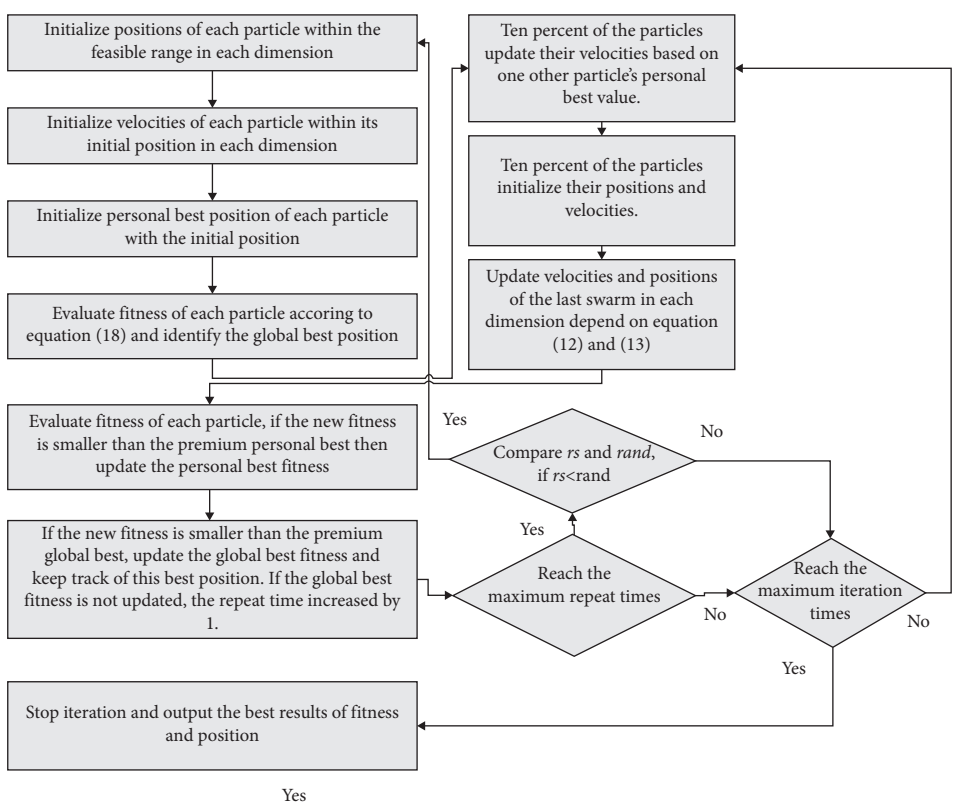


Figure 2: The flowchart of the proposed PSO.

individual and population, speed calculation, and state update are required. Therefore, the time complexity of the proposed method is

$$O(L,K,R,N) = O\left(K * \left(KN + (R-1) * (L-1) * 0.1 * N^2\right)\right) \approx O\left(K^2\right) + O\left(KRLN^2\right). \tag{16}$$

When the size of the problem is much larger than the population size, the time complexity of this method can be simplified as $K * R * L * O(N^2)$; that is, the calculation amount is proportional to the square of the size of the research problem and is proportional to the population size (K), the maximum number of repeats (R), and the maximum number of iterations (L).

4. Experiment Study

An application example is adopted to evaluate the performance of the proposed PSO. The parameters, the constants, and exponents' values are specified as

Type of milling: plain milling.

Motor power: $P_w = 5.5$KW, efficiency $\eta = 0.7$.

Arbor diameter $d_a = 27$mm, arbor length between supports $L_\alpha = 210mm$.

Permissible bending and torsional stress of arbor: $k_b = 140MPa = 14.27\text{kg/mm}^2$, $k_t = 120\text{MPa} = 12.23\text{kg/mm}^2$.

Modulus of elasticity of arbor material $E = 200GPa = 20387\text{kg/mm}^2$.

Spindle speed range: $(31.5 \sim 2000)$ rpm, feed rate range: $(14 \sim 900)$ mm/min.

Tool material: HSS; tool diameter: $D = 63$mm; number of teeth $Z = 8$.

Workpiece material: structural carbon steel $(C \leq 0.6\%)$.

Tensile strength: 750 MPa; Brinell hardness number: 150.

Length, width, depth of cut: $L = 160$mm, $\alpha_w = 50$mm, $\alpha = 5$mm.

Loading and unloading time of one work piece $T_L = 1.5$min.

Set up time of fixtures and machine tool $T_s = 10$min; tool change time $T_c = 5$min.

Process adjusting and quick return time $T_\alpha = 0.1$min/part.

Lot size $N_b = 100$; cutting inclination: 30°.

The constants and exponents used in the example are listed as follows:

$C_v = 35.4$, $m = 0.33$, $b_v = 0.45$, $e_v = 0.3$, $B_m = 1.0$, $B_h = 1$, $B_p = 0.8$, $B_t = 0.8$, $u_v = 0.4$, $r_v = 0.1$, $n_v = 0.1$, $q_v = 0$, $C_{zp} = 68.2$, $b_z = -0.86$, $e_z = 0.86$ and $u_z = 0.72$

$\alpha_{\min}$ and $\alpha_{\max}$ are 0.5 mm, 4 mm for the milling operation, so $0.5 \leq \alpha_i \leq 4$

Calculate the boundary of f_z (feed per teeth) and V (feed speed) based on (13) and (14). The following equations are obtained:

$$0.00875\left(\frac{\text{mm}}{\text{tooth}}\right) \leq f_z \leq 3.571\left(\frac{\text{mm}}{\text{tooth}}\right),$$
$$6.234\left(\frac{m}{\text{min}}\right) \leq V \leq 395.84\left(\frac{m}{\text{min}}\right). \tag{17}$$

The maximum iteration number is set as 40, the population of particles is 20, inertia weight (ω) factor is 0.9, acceleration coefficients c_1, c_2 are 1.05, and the punishment factor δ is 50. All the hyperparameters, such as the c_1 and c_2, are selected based on preliminary experiments, and the experiment results are not sensitive with a feasible selection range. Moreover, the preliminary experiment also suggests that the hyperparameter in this experiment will provide an acceptable result as well. But if it wants to exploit the best performance of the proposed method, a fined search for the hyperparameter is more feasible.

For this simplified two-dimension problem, the width of each dimension is (3.571–0.000875) and (395.84–6.234). It can be seen that the two ranges differ so much. So, the initial speeds are assigned with the initial position coordinates of particles rather than be set at the 10~20% of the dynamic range traditionally. The searchability of particles is balanced in two dissimilar dimensions by the recommended assignment.

Four typical machining strategies with different N_p and α_i are chosen. And the different strategies to accomplish the total 5 mm removal are tabulated in Table 2.

Some other methods are implemented to have a comparison with the proposed PSO. Geometric programming (GP) is one of the typical traditional dynamic programming methods. Artificial bee colony (ABC) and simulated annealing (SA) are kinds of soft computing methods that are researched mostly in recent years. All the comparison methods are the advanced methods of multipass milling process, and for a fair comparison, the proposed improved PSO (IPSO) is compared with these methods directly without any extra design or strategies. The purpose of the comparison is to evaluate if the proposed method is effective for multipass milling process.

Table 3 shows the four groups of computational results of IPSO and ABC. IPSO is superior to ABC in all four strategies. The production time of one part is shorted by 0.038 minutes, especially in machining strategy 1. The total machining time of a batch is shortened by $0.038 \times 100 = 38$ minutes.

The computational results obtained by GP are tabulated in Table 4. Also, Table 4 gives the results of SA in the commonly used strategy 2. The final production time demonstrates the superiority of IPSO to GP and SA. IPSO cut the production time of one part by 0.031 minutes contrasted to SA. And for GP, production time decreases by almost one minute.

IPSO has a self-adapting feature that the particles follow the local best and global best position by adjusting their velocities. The two best positions are updated by fitness estimation and affect the direction of swarms in the next iteration. Particles continually gather to the near-optimal area by conducting two ongoing comparisons. IPSO has efficient search ability due to its active inner learning structure.

ABC gains results approximate to IPSO in Table 3 but ABC algorithm is so complicated in the concept of foragers, and unemployed foragers consist of scout bee and recruit, employed foragers, and experienced foragers [12, 30]. SA is a kind of probabilistic hill-climbing computing algorithm and is suitable for combinatorial optimization problems or nonlinear response functions. According to Table 4, the results obtained by SA are inferior to the results gained by IPSO. Moreover, SA is time-consuming. With about 100 iterations, SA can obtain the final best results [5]. Unlike IPSO, SA uses Metropolis strategy to update the current best solution and only when the annealing time is long enough, SA can provide good results. When compared to GP which is

Table 2: Four machining strategy options.

Machining strategy	1	2	3	4
Roughing operation	$\alpha_{rough} = 2$ $\alpha_{rough} = 2$	$\alpha_{rough} = 1.5$ $\alpha_{rough} = 1.5$ $\alpha_{rough} = 1.5$	$\alpha_{rough} = 2$ $\alpha_{rough} = 1$ $\alpha_{rough} = 1$	$\alpha_{rough} = 1$ $\alpha_{rough} = 1$ $\alpha_{rough} = 1$ $\alpha_{rough} = 1$
Finishing operation	$\alpha_{finish} = 1$	$\alpha_{finish} = 0.5$	$\alpha_{finish} = 1$	$\alpha_{finish} = 1$

Table 3: Computational results of PSO and ABC.

Method		Results of IPSO				Results of ABC				
S. no.	Machining strategy	f_z (mm/tooth)	V (m/min)	T_2 (min)	**Production time** $T_1 + T_2$	f_z (mm/tooth)	V (m/min)	T_2 (min)	**Production time** $T_1 + T_2$	T_1 (min)
1	$\alpha_{rough} = 2$ $\alpha_{rough} = 2$ $\alpha_{finish} = 1$	0.242 0.242 0.190	53.53 53.53 72.61	0.456 0.456 0.428	**3.24**	0.231 0.231 0.189	48.117 48.117 74.090	0.475 0.475 0.428	**3.278**	1.9
2	$\alpha_{rough} = 1.5$ $\alpha_{rough} = 1.5$ $\alpha_{rough} = 1.5$ $\alpha_{finish} = 0.5$	0.341 0.341 0.341 0.435	50.86 50.86 50.86 64.19	0.340 0.340 0.340 0.212	**3.232**	0.337 0.337 0.337 0.432	46.982 46.982 46.982 64.41	0.343 0.343 0.343 0.211	**3.240**	2.0
3	$\alpha_{rough} = 2$ $\alpha_{rough} = 1$ $\alpha_{rough} = 1$ $\alpha_{finish} = 1$	0.242 0.554 0.554 0.190	53.53 47.33 47.33 72.61	0.456 0.225 0.225 0.428	**3.334**	0.231 0.552 0.552 0.189	48.117 47.519 47.519 74.090	0.475 0.226 0.226 0.428	**3.355**	2.0
4	$\alpha_{rough} = 1$ $\alpha_{rough} = 1$ $\alpha_{rough} = 1$ $\alpha_{rough} = 1$ $\alpha_{finish} = 1$	0.554 0.554 0.554 0.554 0.190	47.33 47.33 47.33 47.33 72.61	0.225 0.225 0.225 0.225 0.428	**3.428**	0.552 0.552 0.552 0.552 0.189	47.519 47.519 47.519 47.519 74.090	0.226 0.226 0.226 0.226 0.428	**3.432**	2.1

Table 4: Computational results of IPSO, SA, and GP.

Machining strategy		$\mathbf{f_z}$ (mm/tooth)	$\mathbf{V}$ (m/min)	$\mathbf{T_{pr} = T_1 + T_2}$ (min)
PSO	$\alpha_{rough} = 1.5$ $\alpha_{rough} = 1.5$ $\alpha_{rough} = 1.5$ $\alpha_{finish} = 0.5$	0.341 0.341 0.341 0.435	50.86 50.86 50.86 64.19	**3.232**
SA	$\alpha_{rough} = 1.5$ $\alpha_{rough} = 1.5$ $\alpha_{rough} = 1.5$ $\alpha_{finish} = 0.5$	0.336 0.336 0.336 0.429	44.633 44.633 44.633 57.23	**3.263**
GP	$\alpha_{rough} = 3$ $\alpha_{finish} = 2$	0.338 0.57	26.4 25.16	**4.205**

one type of traditional optimization strategy, IPSO is better than the results of GP. Traditional methods alone cannot give a robust result of the multiconstraints nonlinear programming optimization problem.

5. Conclusion and Future Work

Process planning of the multipass milling process is crucial for the determination of the cutting parameters. The selection of the passes, depth of cut, cut speed, and feed per teeth have a great impact on the production time greatly. The cost and final quality of products are also affected by the recommended parameters. In this study, the multipass machining process is modeled on basis of the proposed PSO which is one kind of swarm intelligence algorithm. Its search mechanism of going after the local best and global best positions ensures the convergence of particles.

The penalty function method is used as the constraints handling technique. The constraints of the arbor strength, the arbor deflection, and the cutting power are satisfied by the punishment function method. An experimental example is presented to test the proposed IPSO. The computational results of IPSO are superior to the results gained by ABC, SA, and GP. Meanwhile, IPSO has a relatively high convergence rate.

In this study, IPSO is successfully used in milling parameters optimization problems, but it has not been frequently tested in other metal machining parameters optimization problems such as drilling, grinding, and turning. In future work, the searching ability and convergence rate of IPSO will be tested by increasing the dimensions of the optimization problem. Furthermore, for a complicated machining process with different machining operations, this will largely decrease the total process time and increase production. Besides, hybridizing IPSO with other optimization algorithms to enhance its performance is another research direction.

Data Availability

The data are from the related work, and it has been illustrated in the experimental parts.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the National Key Technology Support Program under Grant no. 2015BAF01B04 and Key R&D Program of Huber Province under Grant no. 2021AAB001.

References

[1] Y. Gao, X. Li, X. V. Wang, L. Wang, and L. Gao, "A review on recent advances in vision-based defect recognition towards industrial intelligence," *Journal of Manufacturing Systems*, vol. 62, pp. 753–766, 2022.

[2] L. Ma, N. Li, Y. Guo et al., "Learning to optimize: reference vector reinforcement learning adaption to constrained many-objective optimization of industrial copper burdening system," *IEEE Transactions on Cybernetics*, pp. 1–14, 2021.

[3] Y. Gao, L. Gao, X. Li, and X. Yan, "A semi-supervised convolutional neural network-based method for steel surface defect recognition," *Robotics and Computer-Integrated Manufacturing*, vol. 61, Article ID 101825, 2020.

[4] Y. Zhou, T. Xing, Y. Song et al., "Digital-twin-driven geometric optimization of centrifugal impeller with free-form blades for five-axis flank milling," *Journal of Manufacturing Systems*, vol. 58, pp. 22–35, Jan. 2021.

[5] R. Venkata Rao and P. J. Pawar, "Parameter optimization of a multi-pass milling process using non-traditional optimization algorithms," *Applied Soft Computing*, vol. 10, no. 2, pp. 445–456, 2010.

[6] M. Tolouei-Rad and I. M. Bidhendi, "On the optimization of machining parameters for milling operations," *International Journal of Machine Tools and Manufacture*, vol. 37, no. 1, pp. 1–16, 1997.

[7] P. G. Petropoulos, "Optimal selection of machining rate variables by geometric programming," *International Journal of Production Research*, vol. 11, no. 4, pp. 305–314, 1973.

[8] K. Vijayakumar, G. Prabhaharan, P. Asokan, and R. Saravanan, "Optimization of multi-pass turning operations using ant colony system," *International Journal of Machine Tools and Manufacture*, vol. 43, no. 15, pp. 1633–1639, 2003.

[9] P. Palanisamy, I. Rajendran, and S. Shanmugasundaram, "Optimization of machining parameters using genetic algorithm and experimental validation for end-milling operations," *International Journal of Advanced Manufacturing Technology*, vol. 32, no. 7-8, pp. 644–655, 2007.

[10] V. Gaikhe, J. Sahu, and R. Pawade, "Optimization of cutting parameters for cutting force minimization in helical ball end milling of inconel 718 by using genetic algorithm," *Procedia CIRP*, vol. 77, pp. 477–480, 2018.

[11] Z. G. Wang, Y. S. Wong, and M. Rahman, "Optimisation of multi-pass milling using genetic algorithm and genetic simulated annealing," *International Journal of Advanced Manufacturing Technology*, vol. 24, no. 9-10, pp. 727–732, 2004.

[12] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Applied Soft Computing*, vol. 8, no. 1, pp. 687–697, 2008.

[13] A. Baykasoğlu, "Optimising cutting conditions for minimising cutting time in multi-pass milling via weighted superposition attraction-repulsion (WSAR) algorithm," *International Journal of Production Research*, vol. 59, no. 15, pp. 4633–4648, 2021.

[14] J. Li, Y. Sun, and S. Hou, "Particle swarm optimization algorithm with multiple phases for solving continuous optimization problems," *Discrete Dynamics in Nature and Society*, vol. 2021, pp. 1–13, 2021.

[15] L. Ma, M. Huang, S. Yang, R. Wang, and X. Wang, "An adaptive localized decision variable analysis approach to large-scale multiobjective and many-objective optimization," *IEEE Transactions on Cybernetics*, vol. 52, no. 7, pp. 6684–6696, 2022.

[16] B. Song, Z. Wang, and L. Zou, "An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve," *Applied Soft Computing*, vol. 100, Article ID 106960, 2021.

[17] J. Zeng, B. Roy, D. Kumar et al., "Proposing several hybrid PSO-extreme learning machine techniques to predict TBM performance," *Engineering with Computers*, 2021.

[18] Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, vol. 1, pp. 81–86, Seoul, Korea (South), May 2001.

[19] L. Han, K. Xing, X. Chen, and F. Xiong, "A Petri net-based particle swarm optimization approach for scheduling deadlock-prone flexible manufacturing systems," *Journal of Intelligent Manufacturing*, vol. 29, no. 5, pp. 1083–1096, 2018.

[20] X. Tang, C. Shi, T. Deng, Z. Wu, and L. Yang, "Parallel random matrix particle swarm optimization scheduling algorithms with budget constraints on cloud computing systems," *Applied Soft Computing*, vol. 113, Article ID 107914, 2021.

[21] Z. Yang, C. Liu, X. Wang, and W. Qian, "An improved multiobjective PSO for the scheduling problem of panel block construction," *Discrete Dynamics in Nature and Society*, vol. 2016, Article ID 5413520, 13 pages, 2016.

[22] Y. Sun and H. Ren, "A GD-PSO algorithm for smart transportation supply chain ABS portfolio optimization," *Discrete Dynamics in Nature and Society*, vol. 2021, Article ID e6653051, 9 pages, 2021.

[23] B. R. C. Karuppanan and M. Saravanan, "Optimized sequencing of CNC milling toolpath segments using metaheuristic algorithms," *Journal of Mechanical Science and Technology*, vol. 33, no. 2, pp. 791–800, 2019.

[24] R. Naik and N. Sathisha, "Desirability function and GA-PSO based optimization of electrochemical discharge micro-machining performances during micro-channeling on silicon-wafer using mixed electrolyte," *Silicon*, 2022.

[25] Y. Fang, L. Zhao, P. Lou, and J. Yan, "Cutting parameter optimization method in multi-pass milling based on improved adaptive PSO and SA," *Journal of Physics*, vol. 1848, no. 1, Article ID 012116, 2021.

[26] Y. Fan, P. Wang, A. A. Heidari, H. Chen, T. Hamza, and M. Mafarja, "Random reselection particle swarm optimization for optimal design of solar photovoltaic modules," *Energy*, vol. 239, Article ID 121865, 2022.

[27] A. İ. Sönmez, A. Baykasoğlu, T. Dereli, and İ. H. Fılız, "Dynamic optimization of multipass milling operations via geometric programming," *International Journal of Machine Tools and Manufacture*, vol. 39, no. 2, pp. 297–320, 1999.

[28] W. a. Yang, Y. Guo, and W. Liao, "Multi-objective optimization of multi-pass face milling using particle swarm intelligence," *International Journal of Advanced Manufacturing Technology*, vol. 56, no. 5-8, pp. 429–443, 2011.

[29] H. Liu, Z. Cai, and Y. Wang, "Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization," *Applied Soft Computing*, vol. 10, no. 2, pp. 629–640, 2010.

[30] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

Hindawi
Discrete Dynamics in Nature and Society
Volume 2022, Article ID 3635073, 14 pages
https://doi.org/10.1155/2022/3635073

Research Article

Comparative Study of Swarm-Based Algorithms for Location-Allocation Optimization of Express Depots

Yong-Wei Zhang, Qin Xiao, Xue-Ying Sun, and Liang Qi

College of Electronics and Information, Jiangsu University of Science and Technology, Zhenjiang 202003, China

Correspondence should be addressed to Qin Xiao; xiaoqincn@just.edu.cn

Received 11 May 2022; Revised 12 June 2022; Accepted 22 June 2022; Published 29 August 2022

Academic Editor: Shi Cheng

The location and capacity of express distribution centers and delivery point allocation are mixed-integer programming problems modeled as capacitated location and allocation problems (CLAPs), which may be constrained by the position and capacity of distribution centers and the assignment of delivery points. The solution representation significantly impacts the search efficiency when applying swarm-based algorithms to CLAPs. In a traditional encoding scheme, the solution is the direct representation of position, capacity, and assignment of the plan and the constraints are handled by punishment terms. However, the solutions that cannot satisfy the constraints are evaluated during the search process, which reduces the search efficiency. A general encoding scheme that uses a vector of uniform range elements is proposed to eliminate the effect of constraints. In this encoding scheme, the number of distribution centers is dynamically determined during the search process, and the capacity of distribution centers and the allocation of delivery points are determined by the random proportion and random key of the elements in the encoded solution vector. The proposed encoding scheme is verified on particle swarm optimization, differential evolution, artificial bee colony, and powerful differential evolution variant, and the performances are compared to those of the traditional encoding scheme. Numerical examples with up to 50 delivery points show that the proposed encoding scheme boosts the performance of all algorithms without altering any operator of the algorithm.

1. Introduction

Since 2014, the express delivery volume in China has ranked first in the world for six consecutive years [1]. Statistics show 107.7 billion pieces of delivery in China in 2021, with a 31% growth compared with 2020 [2].

The rapid growth of the Express Delivery Industry has brought fierce competition among the participants. The primary delivery companies actively improve service quality and delivery efficiency. For regional express delivery service suppliers, the location of the service center has a significant impact on efficiency and, hence, the overall operation cost.

Such a problem is typically modeled as a logistic distribution problem, which minimizes total distances between the distribution center (DC) and associated delivery points (DPs) under certain constraints. The problem is twofold: one considers the location-allocation problem (LAP) of the distribution centers and the other considers the vehicle routing problem (VRP) that starts at the distribution center and goes through each delivery point and back to the distribution center. The joint problem is the location and routing problem (LRP) [3]. Both LAP and VRP are NP-hard, and the combined LRP has attracted attention since the 1960s [4]. The problem structure of LRP is shown in Figure 1.

In LRP, the optimal route associated with a given distribution center may change with the demands of the delivery point or the policy of distribution centers. Therefore, the VRP may be left to the distribution centers to solve. Since the delivery points for each distribution center are not many, the local VRP can be solved easily. Therefore, the separation of LAP and VRP could significantly reduce the complexity of the problem.

The LAP may apply to a broader region, such as a country, state, or province. The options for locations are limited to a list of lower-level cities, and the options for allocation are subsets of delivery points. The solution of

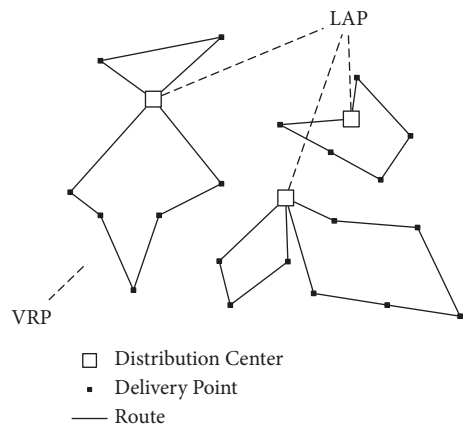


Figure 1: Problem structure of LAP, VRP, and LRP.

location usually uses binary representation because the number of options is fixed. The solution of allocation usually uses integers to denote the belonging of delivery points. Since the total number of distribution centers and delivery points is fixed, the size of a solution can be fixed as well. However, for a city-level LAP, the model and solution representation may vary because the available options for distribution centers can be many, and the setup cost and capacity of different options have a significant impact on the total cost.

On the other hand, the NP-hard nature of LAP makes heuristic algorithms favorable, among which swarm-based algorithms are representative [3, 5, 6]. The solution representation plays a vital role in applying the algorithm because an efficient encoding scheme may reduce the length of the solution string, smooth the solution space, or eliminate the constraints. For example, choosing N distribution centers out of 100 candidate locations requires 100 binary bits to represent the solution, whereas a floating number representation of the coordinates needs only $2N$ numbers. When considering the capacity of a distribution center, the binary representation quantizes the solution space and loses smoothness. Finally, the LAP constraints may result in many infeasible solutions, hence reducing the search efficiency.

This paper discusses the city-level express distribution center location problem and provides a new perspective for simplifying the LAP. Floating numbers denote the coordinates of distribution centers, and the delivery point assignment for each distribution center is determined by the decoded sequence of delivery points and the capacity of the distribution center. The total distance between the distribution center and delivery points, setup costs, and operational costs is used to evaluate the solution. Representative swarm-based algorithms, including particle swarm optimization (PSO), differential evolution (DE), artificial bee colony (ABC), and a powerful variant of DE, and LSHADE-*cn*EpSin [7], are compared under two solution encoding schemes.

2. Literature Review

The LAP tries to determine the location of distribution centers and the delivery points assigned to each distribution center simultaneously. LAP can take various forms in different scenarios. For example, distribution centers are sites for distributing medical services in public health emergencies, and delivery points are affected in literature [8]. Distribution centers are wastewater treatment plants in the wastewater treatment problem, and delivery points are processing units in literature [9]. Distribution centers are web servers in web services, and delivery points are user centers in literature [10]. For an express distribution center location problem, the operational cost involves the capacity and location of each distribution center, and the capacity is either determined or constrained by the assigned delivery points. Such a problem is called a capacitated location-allocation problem (CLAP) [11].

Much effort has been put into this issue. Pham et al. [12] applied a hybrid of the Fuzzy-Delphi-TOPSIS approach to identify the critical criteria for choosing the logistic distribution center. Yang et al. [13] considered the distances between manufactory and distribution centers and between the distribution center and customers and combined tabu search and genetic algorithm to select four distribution centers out of ten candidates. Karaoglan et al. [14] modeled the LRP with simultaneous pickup and delivery, which reflects the practice of beer distribution and empty bottle collection. The problem is then solved using an improved version of the simulated annealing (SA) algorithm.

The solution of CLAP may benefit from a geospatial information system (GIS) since the regional division and distance between two points can be more precise. Vafaeinejad et al. [15] developed a vector assignment ordered capacitated median problem (VAOCMP) model to describe the fire station location and allocation problem. In the VAOCMP model, the arrival time of the fire engine to demand points and the capacity of the fire facility are considered. The closeness of the fire facilities ranks the demand, and a facility will be filled up with closer demands. The problem is then solved by tabu search and simulated annealing (SA). Zheng et al. [16] proposed that the underground metro might be used as a complement to the urban logistics system. They utilize GIS to find the shortest path through all the most demanding points. The demanding points are allocated by the Voronoi diagram, which partitions a plane into polygons such that all the points inside a polygon are closest to one of the communities [17].

The demand for delivery points may be stochastic. Expert opinions can be introduced to build the distribution model of the customers with a lack of data. Zhou and Liu [18] used fuzzy numbers to model the customer demand, and the expected cost was used as the minimization goal. The expected cost is obtained by fuzzy simulation, and the model is solved by network simplex programming and genetic algorithms. The location of the demands may also be stochastic. Mousavi and Niaki [19] used the normal distribution to model demand location and fuzzy variables to model the amount of demand. Three cost functions were proposed: (1) minimization of the fuzzy expected cost, which is the integration of the credibility of fuzzy events; (2) the β-cost minimization model, which minimizes the upper bound of transportation cost that has credibility greater than β; and (3) the credibility maximization model. The model is then solved by using fuzzy simulation and a genetic algorithm.

Noticing the redundancy of express terminal nodes that different express service suppliers establish in the same city, Meng et al. [20] proposed the express terminal nodes optimization integration problem (ETNOIP). The goal of ETNOIP is to establish the minimum number of shared express terminal nodes that could serve a given number of customer clusters. The capacity and scope of an express terminal are included in the cost as well. The model is then solved by SA with neighbor search and shows advantages over immune genetic algorithm (IGA) and CPLEX, an IBM optimization solver.

Many swarm-based algorithms and their variations have emerged in the last two decades. Many swarm-based algorithms have been applied to LAP as well. Xu et al. [21] used the wolf-pack algorithm to optimize the total distance. Bao et al. [6] applied particle swarm optimization (PSO) to a logistic vehicle routing problem. A supported vector machine was introduced to distinguish the state of a particle, and the state will determine whether or not a group of particles will be updated. Moonsri et al. [5] discussed the poultry logistics planning problem, which routes vehicles to each established depot. A new mutation formula is developed in the reinitialization phase of differential evolution (DE) to protect the local structure of the solution, and a location search of partial variables is used to enhance the exploitation ability. Guo and Zhang [3] considered the vehicle routing problem and location-allocation problem as a whole and applied a discrete artificial bee colony (ABC) to determine the choice of recycling centers, the vehicles that serve the recycling center, and the route of the vehicles.

Various models have been proposed in the past decades to describe different scenarios, and more algorithms have been developed to solve the proposed models. The NP-hard nature of CLAP and the constraints that come with it make algorithms unable to reach their full potential. Taking the express distribution center location-allocation problem as an example, we propose a general encoding method for swarm-based algorithms, which eliminates the capacity constraint in allocating delivery points and improves the efficiency of the compared algorithms.

3. Swarm-Based Algorithms for LAP

3.1. Particle Swarm Optimization (PSO). PSO is the most representative swarm-based algorithm. The core mechanism is defined as follows:

$$\begin{aligned} x_{i,j}^{t+1} &= x_{i,j}^{t} + v_{i,j}^{t+1}, \\ v_{i,j}^{t+1} &= w(t)v_{i,j}^{t} + c_1 \mathrm{rand}()\left(x_{i,j}^{g\mathrm{best}} - x_{i,j}^{t}\right) + c_2 r()\left(x_{i,j}^{p\mathrm{best}} - x_{i,j}^{t}\right). \end{aligned} \tag{1}$$

The position of particles represents the candidate solutions. The symbol $x_{i,j}^{t+1}$ denotes the j-th coordinate of a particle i in the $t+1$ generation, which is updated by the velocity $v_{i,j}^{t+1}$ associated with each particle. Three parts determine the velocity of a particle in the next generation:

(1) Current velocity is weighted by a linearly decreasing factor $w(t)$.

(2) The difference between the current position and the global best position $x_{i,j}^{g\mathrm{best}}$ is scaled by the social learning factor c_1 and a uniform distribution random number $r\mathrm{and} \in [0,1]$.

(3) The difference between the current position and the personal best position $x_{i,j}^{p\mathrm{best}}$ is scaled by the personal learning factor c_2 and another uniform distribution random number $r\mathrm{and} \in [0,1]$. The last two parts represent social learning and self-learning.

3.2. Differential Evolution (DE). DE is another widely used swarm-based algorithm that uses other solutions in a swarm, instead of the global or personal best, to generate new solutions. Many mutation operators have been proposed in the literature. In this paper, we adopt the "DE/rand/1" strategy as follows:

$$\begin{aligned} v_{i,j}^{t+1} &= x_{r1,j}^{t} + F\left(x_{r2,j}^{t} - x_{r3,j}^{t}\right),\ r1 \neq r2 \neq r3 \neq i, \\ u_{i,j}^{t+1} &= \begin{cases} v_{i,j}^{t+1}, & \text{if } r\mathrm{and}(\bullet) \leq CR \vee j = r\mathrm{and}n(1,D), \\ x_{i,j}^{t+1}, & \text{otherwise.} \end{cases} \end{aligned} \tag{2}$$

The mutated solution $v_{i,j}^{t+1}$ is generated from three solutions that are randomly selected from the swarm and are different from the current solution and each other. The difference between the two is scaled by a factor F and then added to the third solution. Note that only some bits of the selected solution are mutated by probability CR, allowing a subtle modification of the solution. Additionally, $r\mathrm{and}n(1,D)$ is a random natural number in the range $[1,D]$, ensuring that at least one bit is mutated. The candidate solution $u_{i,j}^{t+1}$ is retained if the corresponding cost is better than the current one. Given the differential nature of the mutation operator (2), DE conducts large-scale exploration in the early stages and subtle exploitation in the later stages.

3.3. Artificial Bee Colony (ABC). ABC produces new solutions in three ways that mimic the behaviors of three types of bees: employed, onlooker, and scout bees. The food sources represent the current best solutions. The employed and onlooker bees share the same mutation operator as follows:

$$v_{i,j}^{t+1} = x_{i,j}^{t} + (2r\mathrm{and}(\bullet) - 1)\left(x_{i,j}^{t} - x_{r,j}^{t}\right), r \neq i. \tag{3}$$

The difference between the current solution $x_{i,j}^{t}$ and a randomly selected solution $x_{r,j}^{t}$ is scaled by a random number in [−1,1], and added to the current solution. This mechanism allows employed and onlooker bees to search for the alternative to the current solution. The behavior difference between employed and onlooker bees is that the employed bees make sure each food source is visited once in a cycle. In contrast, the food source i is visited by an onlooker bee with a probability p_i defined as follows:

$$p_i = \frac{\mathrm{fit}_i}{\sum_{n=1}^{n\mathrm{Pop}} \mathrm{fit}_n}, \tag{4}$$

where nPop is the population size and fit_i is the fitness of the food source i. The onlooker bees are dispatched based on the roulette selection–the solutions with better cost have a higher chance of being visited. Each food source (solution) has a visiting limit. If there is no better solution found around the current solution after a certain number of visits, the food source is abandoned, and a scout bee is sent to generate a new random solution in the search space.

The mechanism of employed bees maintains the diversity of the swarm; onlooker bees exploit the neighbors for better solutions, and the food source visiting limit ensures that the algorithm will not be stuck on some solutions.

3.4. Ensemble Sinusoidal Parameter Adaptation Incorporated with LSHADE (LDES). The parameter settings of DE partially depend on the problem. Therefore, research on the parameter settings of DE [22] and the adaptive parameters of DE [23] is proposed to tackle this problem. In the research stream of adaptive DE, Zhang and Sanderson [24] proposed a self-adaptive DE, JADE, which generalizes the "DE/current-to-best" mutation strategy to "DE/current-to-p-best" and controls the parameters in a self-adaptive manner. Tanabe and Fukunag [25] proposed a Success-History-Based Adaptive DE (SHADE). As an enhancement to JADE, SHADE utilized a history-based parameter adaptation scheme and ranked third in the real-parameter single objective optimization competition, CEC 2013. Tanabe and Fukunaga [26] later proposed the LSHADE algorithm, which extends the SHADE algorithm with the Linear Population Size Reduction (LPSR). The LPSR of LSHADE reduces the number of function evaluations in the exploitation stage of optimization and further enhances the performance. LSHADE wins the CEC 2014 competition.

Two years later, Awad et al. [7] proposed the LSHADE-EpSin algorithm, which incorporated the ensemble sinusoidal parameter adaptation and became the joint winner of the competition of CEC 2016. One year later, Awad et al. [27] proposed an improved algorithm, LSHADE-cnEpSin, to tackle the problems with high correlation between variables. LSHADE-cnEpSin became the second winner in the competition of CEC 2017. For brevity, we will denote LSHADE-cnEpSin as LDES in the rest of the paper.

4. Problem Formulation and Solution Representation

4.1. Capacitated Location-Allocation Problem. A complete planning scheme of distribution centers includes the number of distribution centers, the location and capacity of each distribution center, and the delivery points that are serviced by each distribution center.

The location of distribution centers affects the delivery mode and distance, hence the efficiency and service quality of the distribution centers. The factors that may affect the location selection of distribution centers may be classified into two classes: natural factors and social factors. Natural factors include the natural conditions, such as mountains and rivers, the conditions of the land, and the distribution of roads [11]. Social factors may include infrastructure, client demand distribution, suppliers, and policies.

The capacity of distribution centers determines whether the demand of the assigned delivery point can be served. The assignment of delivery points determines the transportation cost and the operational cost.

This paper focuses on the transportation, setup, and operational cost to simplify the model and highlight the main factors. To build up the objective functions for the cost, let us define the symbols as shown in Table 1.

4.1.1. Transportation Cost. The transportation costs may be affected by capital, fuel, lubricant, and operational costs. Sahin et al. [28] showed that the total cost of a unit of cargo in road transportation with trucks consists of 14% investment cost, 60% fuel cost, 17% operational cost and maintenance cost of the vehicle, and 9% external cost, which are positively related to the route length. Assuming the vehicles are fully loaded and the cost has a linear relationship with the route length, the total transportation cost C_t can be formulated in the following equation:

$$C_t = \sum_{i}^{N} \% \sum_{j \in \Omega_i} d_{ij}, \tag{5}$$

where d_{ij} is the city block distance as follows:

$$d_{ij} = \left| p_i^{\text{tx}} - p_j^{\text{dx}} \right| + \left| p_i^{\text{ty}} - p_j^{\text{dy}} \right|. \tag{6}$$

4.1.2. Setup Cost. The setup cost may vary depending on how the distribution center is set, such as renting a warehouse or building a new depot. Assuming that the average setup cost for any possible location is known, the total setup cost is simply as follows:

$$C_s = \sum_{i}^{N} T\left(p_i^{\text{tx}}, p_i^{\text{ty}}\right), \tag{7}$$

where the operational cost for a given point $(p_i^{\text{tx}}, p_i^{\text{ty}})$ can be determined in advance through investigation.

4.1.3. Operational Cost. The operational cost depends on how many demands a distribution center needs to meet, which is usually described by a cubic function of demand [29]. Therefore, a distribution center's operational cost per unit demand is a quadratic function of demand. The total operational cost is formulated as follows:

$$\begin{aligned} C_o &= \sum_{i}^{N} a_1 Q_i^3 + a_2 Q_i^2 + a_3 Q_i, \\ Q_i &= \sum_{j \in \Omega_i} r_j, \end{aligned} \tag{8}$$

where $a_k\,(k = 1, 2, 3)$ are polynomial coefficients. The total demand Q_i that is assigned to the distribution center i is

Table 1: Symbols and description.

Description	Symbol	Property
Maximum number of distribution centers	$N_{\max}$	Parameter
Number of distribution centers	N	Decision variable
Number of delivery points	M	Parameter
Horizontal coordinate of distribution center i	$p_i^{\text{tx}}, i \in 1, \ldots, N$	Decision variable
Vertical coordinate of distribution center i	$p_i^{\text{ty}}, i \in 1, \ldots, N$	Decision variable
Maximum horizontal coordinate of distribution center	$p_{\max}^{\text{tx}}$	
Horizontal coordinate of the delivery point j	$p_j^{\text{dx}}, j = 1, \ldots, M$	Parameter
Vertical coordinate of the delivery point j	$p_j^{\text{dy}}, j = 1, \ldots, M$	Parameter
Distance between the i-th distribution center and j-th delivery point	d_{ij}	Parameter
Capacity of distribution center i	$c_i, i \in 1, \ldots, N$	Decision variable
The demand of delivery point j	$r_j, j \in j, \ldots, M$	Parameter
Total demand of all delivery points	R	Parameter
Set of all delivery points	$\Omega = \{1, \ldots, M\}$	Parameter
Set of delivery points served by distribution center i	Ω_i	Decision variable
Average operational cost is determined by the position of the distribution center $(p_i^{\text{tx}}, p_i^{\text{ty}})$	$T(p_i^{\text{tx}}, p_i^{\text{ty}})$	Parameter

determined by the decision variable Ω_i. The cost per unit demand is then formulated as

$$C_{\text{pud}} = a_1 Q_i^2 + a_2 Q_i + a_3. \tag{9}$$

The optimal capacity for a distribution center is the solution to the following equation:

$$\frac{dC_{\text{pud}}}{dQ_i} = 2a_1 Q_i + a_2 = 0. \tag{10}$$

To summarize, the total cost for an express CLAP is

$$C = C_t w_t + C_s w_s + C_o w_o, \tag{11}$$

where w_t, w_s, and w_o are the weight coefficients for transportation, setup, and operational cost, respectively.

4.1.4. Constraints. The boundary constraints for the decision variables are as follows:

$$0 \le N \le N_{\max}, \tag{12}$$

$$0 \le p_i^{tx} \le p_{\max}^{tx}, 0 \le p_i^{ty} \le p_{\max}^{ty}, i \in 1, \ldots, N. \tag{13}$$

The capacity of the distribution center must satisfy the total demand of all delivery points serviced by it:

$$Q_i \le c_i \le R, R = \sum_{j=1}^{M} r_j, \tag{14}$$

where R is the total demand as described in Table 1. Each delivery point is serviced by one distribution center:

$$\Omega = \sum_{i=1}^{N} \Omega_i, \Omega_i \cap \Omega_j = \varnothing \text{ for any } i \ne j. \tag{15}$$

4.2. Encoding Scheme and Evaluation Criteria. The selected algorithms work on a set of floating number vectors $\mathbf{x}_i$ (known as the population $X = [\mathbf{x}_i, \ldots, \mathbf{x}_{n\text{Pop}}]^T$, where nPop is the population size). Therefore, the solutions of CLAP need to be encoded into floating number vectors before the algorithms can be applied. When a new vector is found by an algorithm, it must be interpreted (decoded) into the actual location and allocation plans before it can be evaluated. This section discusses the encoding/decoding schemes and the evaluation criteria of the plan.

4.2.1. Traditional Encoding Scheme. A complete location planning scheme of distribution centers can be represented by the decision variables described in Table 1. The location of a distribution center requires two numbers to denote the horizontal and vertical coordinates. Another quantity is required to denote the capacity. Furthermore, a number denoting the belonging of a delivery point is also required. Given that the length of the solution representation in the selected algorithms is fixed, the traditional encoding/decoding scheme can be as shown in Figure 2.

For a traditional encoding/decoding scheme, the length of a solution string is $3N + M$, where $g_j \in \mathbb{Z}^+, j \in 1, \ldots, N$ is a positive integer and denotes that the j-th delivery point belongs to the g_j-th distribution center. The decision variables are obtained as follows:

$$\begin{aligned} p_i^{\text{tx}} &= x_{3i-2}, \\ p_i^{\text{ty}} &= x_{3i-1}, \\ c_i &= x_{3i}, \\ i &= 1, \ldots, N, \\ g_j &= x_{3N+j}, \\ j &= 1, \ldots, M, \\ \Omega_i &= \left\{ j = 1, \ldots, M | g_j = i \right\}, \end{aligned} \tag{16}$$

where $x_i \in \mathbb{R}^+$ are floating numbers. x_{3i-2} and x_{3i-1} $(i = 1, \ldots, N)$ have the same range as p_i^{tx} and p_i^{ty}, respectively. The range of x_{3i} $(i = 1, \cdots, N)$ is $[0, R]$ because the lower bound of the capacity cannot be determined in this

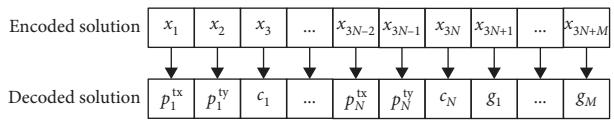


Figure 2: Traditional encoding/decoding scheme TES.

encoding scheme. $x_{3N+j} \in [0, N]$ $(j = 1, \cdots, M)$ and $\lceil \cdot \rceil$ is the rounded-up function.

The traditional encoding scheme is straightforward, but the drawbacks are clear: (1) the number of distribution centers must be determined in advance, and (2) solution strings generated in the search process may violate the constraints. For the boundary constraints defined in Equations (12) and (13), the abovementioned algorithms will truncate the exceeded solutions. However, for the constraints in Equations (12) and (13), an infeasible solution means that the decoded plan cannot be executed.

For example, the delivery points assigned to a distribution center may have greater total demand than the capacity of the distribution center, or no delivery points may be assigned to a distribution center. Hence, a punishment term is defined as follows to suppress the infeasible solutions:

$$C_p = \sum_{i}^{N} \left(\max\left(\sum_{j \in \Omega_i} r_j - c_i, 0 \right) \right). \tag{17}$$

The modified cost function can be as follows:

$$C = C_t w_t + C_s w_s + C_o w_o + C_p w_p, \tag{18}$$

where w_p is the weight of the punishment term. If no delivery point is assigned to a distribution center, the wasted resources are naturally a punishment for the cost. Such solutions will not compete with the solutions that take advantage of the available capacity.

4.2.2. Constraint-Solved Encoding Scheme. The abovementioned encoding scheme and cost function allow the existence of infeasible solutions, which decreases the search efficiency since there are invalid calculations for the infeasible solutions. This paper presents an improved encoding scheme that introduces random proportion and random key (RPK) as shown in Figure 3, which transfers the constraint problem into an unconstraint problem.

As shown in Figure 3, a floating number vector $\mathbf{x} = [x_1, \ldots, x_{3N_{\max}+M+1}]$ is used as the solution string, where $x_i \in [0, 1]$. The number x_1 will be mapped into the number of distribution centers and $[x_2, \ldots, x_{2N+1}]$ will be mapped into the position of each distribution centers by

$$N = x_1 N_{\max}, \tag{19}$$

$$\begin{aligned} p_i^{tx} &= x_{2i} p_{\max}^{tx}, \\ p_i^{ty} &= x_{2i+1} p_{\max}^{ty}, i = 1, \ldots N. \end{aligned} \tag{20}$$

Note that if $N < N_{\max}$, the numbers $[x_{2N+2}, \ldots, x_{2N_{\max}+1}]$ will be omitted.

The numbers $[x_{2N_{\max}+2}, \ldots, x_{3N_{\max}+M+1}]$ can be decoded into the capacity of each distribution center and the assignment of delivery points through random proportion and random key mapping.

The random key mapping determines a sequence of delivery points as follows. The partial vector $[x_{3N_{\max}+2}, \ldots, x_{3N_{\max}+M+1}]$ is sorted in either ascending or descending order to obtain the sorting index $[h_1, \ldots, h_M]$. For example, h_1 denotes the order of x_{3N+2} in the sorted vector and h_M denotes the order of x_{3N+M+1} in the sorted vector. The index $[h_1, \ldots, h_M]$ represents the sequence of the delivery points.

The random proportion mapping uses a partial vector $[x_{2N_{\max}+2}, \ldots, x_{2N_{\max}+N+1}]$ to determine the lower bound capacity $\tilde{c}_i$ of a distribution center:

$$\begin{aligned} \tilde{c}_i &= \frac{x_{2N+1+i}}{\sum_{j=2N+1+i}^{3N+1} x_j} R, \\ i &= 1, \ldots, N-1. \end{aligned} \tag{21}$$

Note that the numbers $[x_{2N_{\max}+N+2}, \ldots, x_{3N_{\max}+1}]$ are omitted as well if $N < N_{\max}$.

The assignment of delivery points for the distribution center i is determined as follows:

$$\begin{aligned} \Omega_i &= \left\{ h_{a(i)}, \ldots, h_{b(i)} \right\}, \\ \tilde{c}_i &\le \sum_{j \in \Omega_i} r_j, i = 1, \ldots, N-1, \end{aligned} \tag{22}$$

$$1 = a(1) < b(1) < a(2) < b(2) < \ldots < a(N) < b(N) = M, \tag{23}$$

where $\left\{ h_{a(i)}, \ldots, h_{b(i)} \right\}$ is a continuous partial sequence in $[h_1, \ldots, h_M]$, which makes the summation of demands from the set of delivery points Ω_i just greater than the lower bound $\tilde{c}_i$.

The actual capacity of the distribution center i is determined by the summation of demand from the delivery point assigned to it:

$$c_i = \sum_{j \in \Omega_i} r_j, i = 1, \ldots, N. \tag{24}$$

The idea of RPK is to determine the number of distribution centers dynamically and make the capacity just enough for the assigned delivery points. The logical procedure of obtaining a feasible capacitated location-allocation plan from any $\mathbf{x} \in [0, 1]^{3N+M+1}$ is as follows:

Step 1: obtain the number of distribution centers N by Equation (19)

Step 2: obtain the position of N distribution centers by Equation (20)

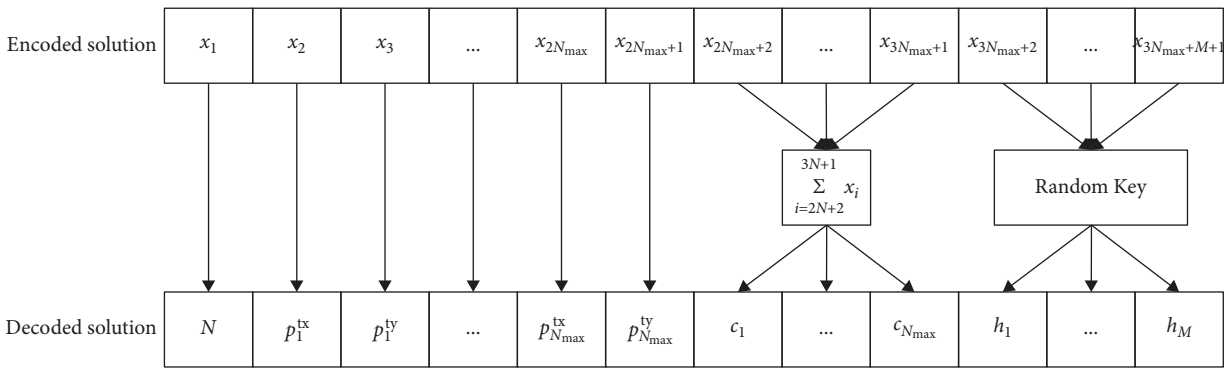


Figure 3: Unconstraint encoding/decoding scheme RPK.

Step 3: sort $[x_{3N_{\max}+2}, \ldots, x_{3N_{\max}+M+1}]$ and get the index $[h_1, \ldots, h_M]$

Step 4: $a(1) \leftarrow 1$

Step 5: for i in $\{1, \ldots, N-1\}$ do

Calculate the lower bound capacity $\tilde{c}_i$ by (21)

Starting from $a(i)+1$, find the first $b(i)$ that satisfies Equation (22)

The set of delivery points assigned to the distribution center i is $\Omega_i = \{h_{a(i)}, \ldots, h_{b(i)}\}$

$a(i+1) \leftarrow b(i)+1$

Obtain the actual capacity of the distribution center i by (24)

Step 6: $\Omega_N = \{h_{b(3)+1}, \ldots, h_M\}$, $c_N = \sum_{j \in \Omega_N} r_j$

A decoding example is as follows. Assume that $N_{\max} = 2$, $M = 5$, $p_{\max}^{tx} = 100$, $p_{\max}^{ty} = 120$, and $R = 500$, and demands for each delivery point are [100, 80, 110, 150, 60]. A vector $\mathbf{x}$ with the length $3N + M + 1 = 12$ is [0.66, 0.1, 0.35, 0.5, 0.6, 0.7, 0.9, 0.4, 0.95, 0.25, 0.55, 0.6]. Then, $N = 0.66 \times 2 = 2$, $p_1^{tx} = 0.1 \times 100 = 10$, $p_1^{ty} = 0.35 \times 120 = 42$, $p_2^{tx} = 0.5 \times 100 = 50$, $p_2^{ty} = 0.6 \times 120 = 72$, $\tilde{c}_1 = 0.7/0.7 + 0.9 \times 500 = 218.75$, and $\tilde{c}_2 = 0.9/0.7 + 0.9 \times 500 = 281.25$. The sorting index of the ascending order of [0.4, 0.95, 0.25, 0.55, 0.6] is [1–5]. The minimum set Ω_1 that satisfies equation (22) is $\{2, 5, 1\}$; therefore, $c_1 = r_2 + r_5 + r_1 = 240$, $\Omega_2 = \{3, 4\}$, and $c_2 = r_3 + r_4 = 260$.

For an arbitrary $\mathbf{x} \in [0, 1]$, the RPK encoding scheme always produces a unique and feasible plan. The total capacity of the distribution centers equals the total demand, which maximizes resource utilization. Furthermore, the RPK encoding scheme allows algorithms to operate on a uniform vector $\mathbf{x} \in [0, 1]$, which facilitates the application of algorithms.

5. Evaluation Experiments

5.1. Experiment Settings. The evaluation experiments were conducted on a 10 km by 10 km square region of Zhenjiang, China [30]. The map of this region is divided into 10×10 grids, and the setup cost for each lattice is obtained via investigation. The map, grid, and setup cost matrix are shown in Figure 4.

The setup costs are scaled into five levels. Level 5 represents the most expensive setup cost. The central area (slightly above the middle) has the highest cost, and the suburbs have the lowest cost. A mountain is located slightly below the middle, and some waters are in the north, where a distribution center cannot be set up. We set the cost much higher than the maximum level cost (20 in this case) to prevent generating a distribution center located in these areas. While a finer grid may bring a plan closer to reality, a 10×10 grid is sufficient to show the algorithm's mechanism and maintain the map's readability.

The TES and RPK encoding schemes are applied to four algorithms: PSO, DE, ABC, and LDES. Since the compared algorithms use different population sizes, we use the number of cost function evaluations (FEs) instead of the number of generations to measure the performance. The algorithms will stop when the maximum FEs are reached. RPK works on a unified search space and transforms a solution $\mathbf{x} \in [0, 1]^D$ into a location and allocation plan, where D is the dimension of the search space, i.e., the number of variables. The algorithm parameters are shown in Table 2.

5.2. Comparison of TES and RPK Encoding Schemes. The comparison is conducted on the map in Figure 4. There are 20 delivery points on the map, each with different demands. The maximum number of delivery centers is set to 5. The location of delivery points and the associated demands are shown in Figure 5.

Each algorithm runs 30 times, and the optimum costs found by each algorithm are averaged over 30 runs. The results are shown in Table 3.

From Table 3, we observe that the RPK encoding scheme improves the performance of all compared algorithms. The average optimum cost was reduced by at least 11.38%. ABC shows the best average performance for both RPK and TES encoding schemes. The DE and LDES show comparable performance with ABC for the RPK encoding scheme, while the smaller standard deviations (0.05 and 0.09), respectively, indicate more stable performance.

The improvement has two sources: first, the RPK allows automatic selection of the number of distribution centers; since each distribution center has a setup cost, fewer

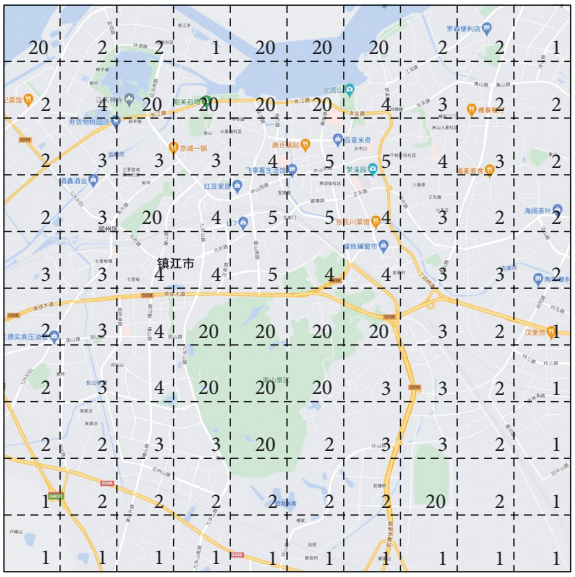


Figure 4: The regional grid map of Zhenjiang, China, with an operational cost for each lattice.

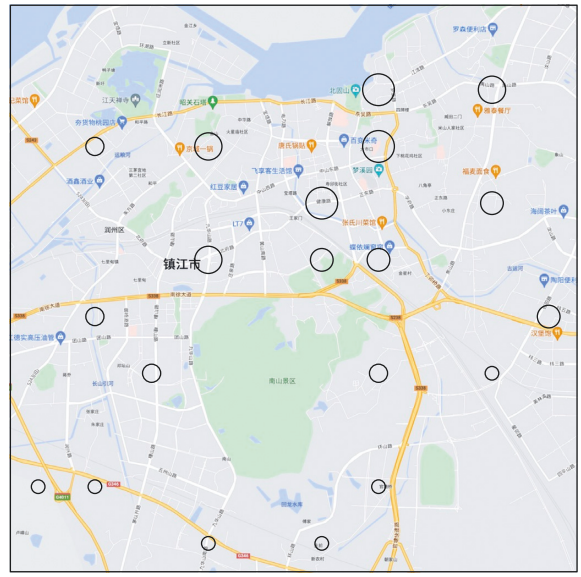


Figure 5: Location and demands of delivery points. The circles show the location, and the circle's diameter shows the value of demand. The larger the diameter, the greater the demand.

Table 2: Parameter settings for PSO, DE, ABC, and LDES algorithms.

Parameter description	Value
General parameters	
The maximum number of cost function evaluations $FE_{\max}$	200,000
Search space	$[0,1]^D$
Transportation cost weight w_t	1
Setup cost weight w_s	1
Operational cost weight w_t	0.1
Punishment weight w_p	30
Polynomial coefficients a for operational cost	$[0.0025, -0.1, 2]$
Coordinate bounds $[p^{tx}_{\max}, p^{ty}_{\max}]$	$[0,1]$
PSO parameters (Bao et al. [6])	
Social learning factor c_1	2
Personal learning factor c_2	2
Weight range $[w_{\min}, w_{\max}]$	[0.4,0.9]
Population size nPop	30
Velocity range $[v_{\min}, v_{\max}]$	[−0.2,0.2]
DE parameters (Moonsri et al. [5])	
Population size nPop	37
Crossover probability	0.9455
Scaling factor	0.6497
ABC parameters (Guo and Zhang [3])	
Population size	50
Number of onlooker bees	25
Number of employed bees	25
Food source visiting limit	500
LDES parameters (Awad et al. [27])	
Initial population size	360
Minimum population size	4
Covariance matrix learning probability pc	0.4
Selection probability ps	0.5
Selection rate of the best solutions p_best_rate	0.11
Archive rate arc_rate	1.4
Memory size	4
The initial number of neighbors SEL	180
Scaling factors	Adaptive adjust

Table 3: Average cost over 30 runs and standard deviate for PSO, DE, ABC, and LDES using RPK and TES encoding scheme.

Encoding scheme	Average cost (standard deviation)			
	PSO	DE	ABC	LDES
RPK	17.23 (0.74)	15.95 (0.05)	**15.54** (0.4)	15.96 (0.09)
TES	21.71 (1.39)	19.35 (0.61)	17.53 (0.52)	18.82 (0.52)
Improvement	20.64%	17.57%	11.38%	15.20%

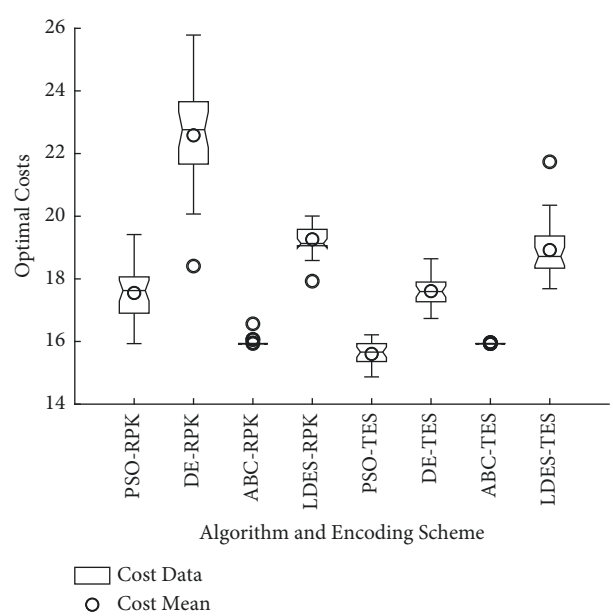


Figure 6: Statistics of the optimal costs in 30 runs of PSO, DE, ABC, and LDES using RPK and TES encoding scheme.

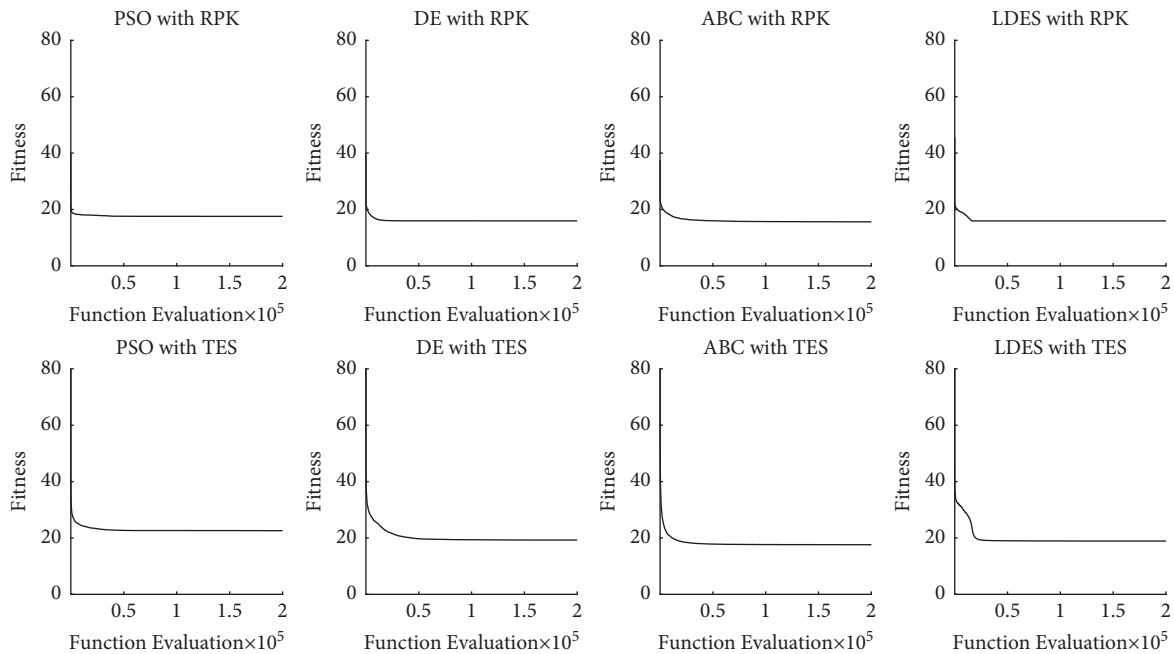
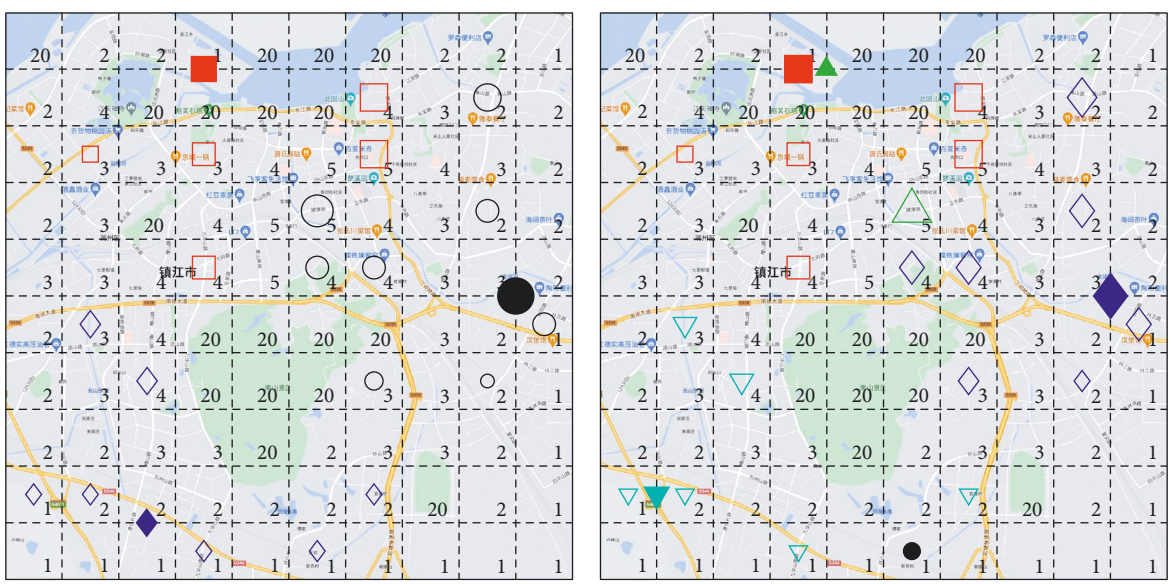


Figure 7: Convergence curves of PSO, DE, ABC, and LDES using RPK and TES encoding scheme.



(a) (b)

Figure 8: Location and allocation plan of ABC using RPK (a) and TES (b) encoding schemes.

distribution center reduces the total cost. However, fewer distribution centers increase the number of delivery points to be serviced, hence increasing the route length of travel through all delivery points and leading to greater transportation costs. On the other hand, TES adopts a fixed number of distribution centers (in this case, 5), and the setup cost is on an average greater than RPK. The RPK balances the number of distribution centers and the route length, which allows each distribution center to operate at a lower average cost.

Second, the capacity distribution mechanism in RPK ensures that the total capacity equals the total demand and no capacity is wasted, which produces a more efficient allocation plan.

The statistics and convergence curves of the four algorithms in the two encoding schemes are shown in Figures 6 and 7. The nonoverlapping notch of the boxes corresponding to different encoding schemes of the same algorithm shows that RPK reduces the median of the optimal cost by 5%.

Table 4: The detailed location and allocation plan of ABC using RPK and TES encoding schemes.

Encoding	DC index	p_i^{tx}	p_i^{ty}	c_i	Ω_i	Q_i	$C_{t,i}$	$C_{s,i}$	$C_{o,i}$
RPK	DC #1	0.900	0.500	24.000	[3, 5–7, 9, 15, 18, 19]	24	2.501	1	24.960
	DC #2	0.350	0.100	20.000	[2, 4, 11, 14, 20]	20	1.651	1	20.000
	DC #3	0.249	0.900	9.000	[1, 8, 10, 12, 13, 16, 17]	9	2.051	1	11.723
	Subtotal	N/A	N/A	53.000	N/A	53	6.203	3	56.683
TES	DC #1	0.550	0.951	1.071	17	1	0.001	1	2.031
	DC #2	0.350	0.100	20.001	[2, 4, 11, 14, 20]	20	1.650	1	20.001
	DC #3	0.900	0.500	19.054	[3, 6, 7, 9, 15, 18, 19]	19	2.001	1	19.097
	DC #4	0.399	0.100	5.004	5	5	0.401	1	7.817
	DC #5	0.100	0.850	8.000	[1, 8, 10, 12, 13, 16]	8	1.700	1	10.880
	Subtotal	N/A	N/A	53.130	N/A	53	5.753	5	59.825

Table 5: The averaged optimal costs and standard deviations with various numbers of DCs and DPs.

Number of DCs	Algorithm	Encoding scheme	Number of DPs				
			#DP = 10	#DP = 20	#DP = 30	#DP = 40	#DP = 50
#DC = 2	PSO	RPK	9.75 (0.2)	17.71 (1.1)	28.93 (0.9)	63.35 (1.1)	100.94 (0.9)
		TES	10.00 (0.6)	19.14 (0.8)	31.29 (1.1)	66.06 (1.3)	104.53 (1.6)
	DE	RPK	9.79 (0.0)	16.18 (0.4)	27.11 (0.3)	60.78 (0.3)	97.12 (0.6)
		TES	9.17 (0.3)	16.65 (0.6)	27.39 (0.5)	61.36 (0.7)	102.68 (3.7)
	ABC	RPK	9.00 (0.2)	15.94 (0.0)	27.07 (0.2)	60.52 (0.1)	97.19 (0.2)
		TES	9.16 (0.3)	16.76 (0.4)	28.22 (0.6)	62.24 (0.7)	99.26 (0.9)
	LDES	RPK	9.76 (0.2)	15.94 (0.0)	27.04 (0.3)	60.90 (0.3)	97.11 (0.3)
		TES	8.87 (0.1)	16.12 (0.2)	28.47 (0.7)	63.70 (0.9)	102.22 (1.0)
#DC = 4	PSO	RPK	9.74 (0.2)	17.20 (0.9)	25.67 (1.1)	36.72 (1.7)	46.76 (1.7)
		TES	12.38 (0.8)	21.14 (1.3)	29.44 (1.9)	42.22 (1.6)	54.54 (2.1)
	DE	RPK	9.79 (0.0)	15.95 (0.0)	21.69 (0.7)	34.98 (2.5)	44.31 (2.1)
		TES	12.09 (0.5)	17.95 (0.4)	24.19 (1.3)	37.95 (2.6)	48.81 (3.4)
	ABC	RPK	8.94 (0.2)	15.59 (0.3)	21.43 (0.4)	30.07 (0.6)	38.86 (0.6)
		TES	10.85 (0.3)	16.68 (0.5)	22.56 (0.6)	32.68 (0.9)	42.24 (1.3)
	LDES	RPK	9.75 (0.1)	15.94 (0.0)	22.01 (0.4)	31.75 (1.3)	41.42 (1.4)
		TES	10.99 (0.3)	17.74 (0.5)	24.52 (1.0)	35.54 (1.5)	46.47 (1.6)
#DC = 6	PSO	RPK	9.76 (0.1)	17.74 (0.9)	25.53 (1.2)	36.94 (1.5)	45.41 (1.6)
		TES	15.42 (1.2)	23.72 (1.6)	33.20 (1.9)	44.28 (2.5)	53.08 (2.3)
	DE	RPK	9.79 (0.0)	15.94 (0.0)	21.44 (0.5)	34.33 (2.7)	41.59 (3.0)
		TES	14.25 (0.8)	20.49 (0.6)	26.99 (1.4)	40.48 (3.5)	51.23 (5.3)
	ABC	RPK	8.92 (0.2)	15.53 (0.4)	21.55 (0.4)	30.11 (0.6)	36.45 (1.1)
		TES	12.72 (0.2)	18.46 (0.4)	23.88 (0.5)	31.47 (0.8)	37.29 (1.0)
	LDES	RPK	9.79 (0.0)	15.94 (0.0)	22.21 (0.7)	31.52 (1.6)	39.23 (2.4)
		TES	13.19 (0.5)	20.39 (0.9)	26.53 (1.1)	35.21 (1.3)	41.94 (2.1)
#DC = 8	PSO	RPK	9.73 (0.2)	17.31 (0.9)	25.66 (1.3)	37.13 (1.5)	45.11 (1.8)
		TES	17.94 (1.2)	27.92 (1.9)	37.98 (2.1)	48.02 (3.0)	56.36 (3.3)
	DE	RPK	9.78 (0.0)	15.96 (0.0)	23.17 (1.1)	34.54 (1.9)	43.21 (3.8)
		TES	15.37 (0.5)	22.49 (0.5)	36.58 (2.5)	48.52 (10.8)	58.75 (15.8)
	ABC	RPK	8.94 (0.2)	15.54 (0.4)	21.49 (0.4)	30.32 (0.5)	36.47 (0.7)
		TES	14.76 (0.2)	20.41 (0.4)	25.98 (0.5)	32.75 (0.6)	37.29 (0.8)
	LDES	RPK	9.79 (0.0)	15.96 (0.1)	22.27 (0.7)	31.10 (1.5)	38.72 (2.7)
		TES	15.73 (0.7)	22.22 (0.8)	29.23 (2.2)	36.24 (1.3)	42.00 (2.1)
#DC = 10	PSO	RPK	9.76 (0.1)	17.86 (1.1)	25.67 (1.7)	36.81 (1.3)	45.10 (1.8)
		TES	22.21 (1.9)	31.67 (3.0)	41.91 (3.3)	52.69 (3.1)	60.96 (3.4)
	DE	RPK	9.78 (0.1)	17.02 (0.7)	24.45 (1.3)	33.70 (2.0)	40.60 (3.1)
		TES	17.60 (0.7)	24.81 (0.8)	39.60 (3.1)	57.69 (14.9)	82.14 (27.3)
	ABC	RPK	9.07 (0.2)	15.47 (0.4)	21.66 (0.7)	30.33 (0.7)	36.75 (0.9)
		TES	16.74 (0.2)	22.40 (0.5)	27.97 (0.5)	34.60 (0.5)	39.06 (0.8)
	LDES	RPK	9.69 (0.3)	15.95 (0.0)	22.62 (0.9)	31.09 (1.9)	38.77 (2.5)
		TES	17.40 (0.5)	23.96 (0.9)	31.38 (2.5)	38.13 (1.3)	45.89 (4.5)

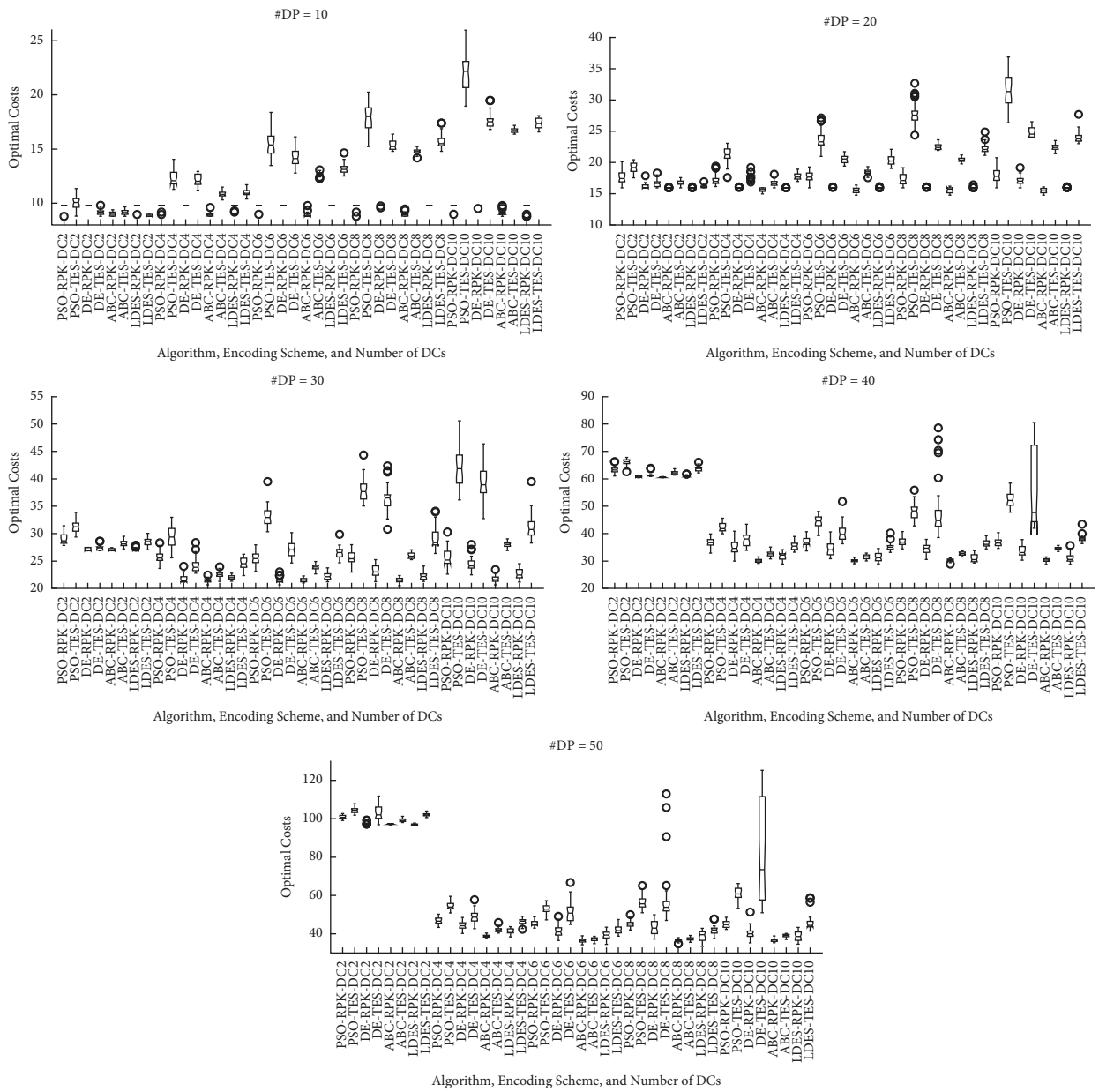


Figure 9: Boxplots of the performance statistics for different combinations of algorithms, encoding schemes, number of DCs, and number of DPs.

Figure 7 shows that most optimization processes converge within 50000 function evaluations regardless of the algorithm or the encoding scheme. Except for ABC with RPK, the search of the other algorithms makes small progress throughout the entire process. This observation shows that the visiting limit mechanism keeps the ABC from being stuck in the local optima.

The best location and allocation plan found by ABC using RPK and TES encoding schemes are shown in Figure 8. A distribution center and associated delivery points are depicted in the markers with the same shape, where a solid marker denotes the distribution center and hollow markers denote delivery points. The size of each marker is proportional to the capacity/demand of DC/DP.

Both RPK and TES avoid the mountains and water areas. RPK allows the algorithm to automatically choose the number of distribution centers (in this case, 3). The delivery points are clustered around each distribution center to minimize transportation costs. The optimized number of distribution centers distributes the demands so that each distribution center may operate at a capacity with the lowest possible cost. For the TES, a fixed number of distribution centers prevents some distribution centers from operating at the lowest possible cost. Although the average transportation cost is reduced due to fewer delivery points serviced by each distribution center, the increased setup cost and operational cost make the allocation plan less economical.

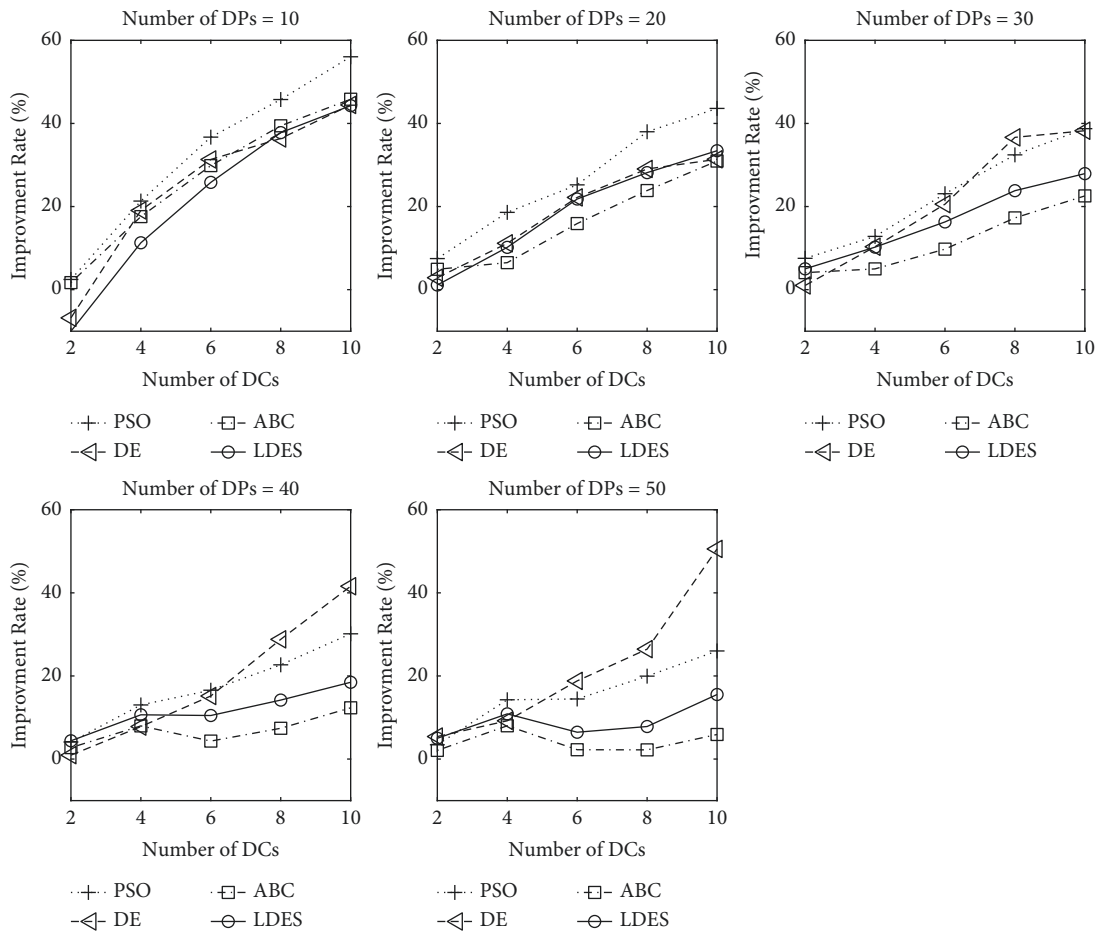


Figure 10: Improvement of RPK over TES when the number of DPs increases.

The algorithm is smart enough under both encoding schemes to choose a location right on the edge of the lattice, which minimizes the setup cost and the transportation cost simultaneously. The detailed plan is shown in Table 4. The symbols $C_{t,i}$, $C_{s,i}$, and $C_{o,i}$ represent the transportation, setup, and operational costs of each distribution center, respectively.

The coordinates p_i^{tx} and p_i^{ty} show that all distribution centers are located on the edge of the lattices, where the setup cost is the lowest, and the distance to delivery points is minimized. The capacities of each distribution center with the RPK encoding scheme are the same as the total demands assigned to them. In contrast, the capacities of TES are slightly greater than the demands, which are a waste of resources and cause greater costs. Meanwhile, the capacities of RPK are closer to the optimal capacity (the optimal capacity can be obtained by solving (10), which is 20 in this case). The total transportation cost of RPK is greater than TES. However, lower setup costs and operational costs compensate for the overall cost.

5.3. Sensitivity of the Number of DCs and DPs. This section considers the impact of the number of distribution centers and delivery points. The number of delivery points is 10, 20, 30, 40, and 50. The maximum number of distribution centers is 2, 4, 6, 8, and 10. All four algorithms are tested. RPK and TES are also compared. The results are given in Table 5.

Improvement of RPK over TES can be observed in most cases (except for DE and LDES with #DP = 10 and #DC = 2). For #DC = 2 cases, the improvements of RPK with different numbers of delivery points are not significant because two distribution centers are not enough for any encoding scheme to distribute the demands into economic capacity. When #DC = 4, the average improvement is at least 5% (ABC with #DP = 30), and the highest improvement is 21.3% (PSO with #DP = 10). When #DC = 10, the average improvement is up to 56.05% (PSO with #DP = 10).

For #DP = 10, the averaged costs of RPK are steady with different numbers of distribution centers, whereas the averaged costs of TES keep rising with the increasing number of distribution centers. The reason behind this observation is that two distribution centers are sufficient for the optimal distribution of the total demand of 10 delivery points. Even if the maximum number of distribution centers varies, RPK automatically selects two distribution centers to distribute the demands and produce similar solutions. On the other hand, TES uses a fixed number of distribution centers. Therefore, the TES has a similar performance with RPK

when #DC = 2 but deteriorated performance with the increasing number of distribution centers because additional distribution centers cause additional setup and operational costs.

The statistics of different combinations of algorithms, encoding schemes, and the number of distribution centers are shown in Figure 9.

The improvements of RPK over TES are shown in Figure 10. When the total demands rise with the number of delivery points, the maximum number of distribution centers that RPK shows a significant improvement (over 20%) rises as well. For example, when #DP = 10, the maximum number of distribution centers needs to be at least 4 for PSO to have an improvement of greater than 20%. When #DP = 50, the number rises to 10. Below a certain maximum number of distribution centers, neither RPK nor TES could find a better allocation plan. In contrast, above the threshold, the dynamic number of distribution centers in RPK shows excellent efficiency in solving the CLAP.

6. Conclusion

The solution representation of practical engineering problems may significantly affect the performance of swarm-based algorithms. Proper encoding of the solutions may bring three significant advantages:

(1) The encoded solution could have uniform ranges, which is suitable for the algorithm adopting a "crossover" operator that may switch the position of the elements in a solution vector

(2) The landscape of the solution space is altered to provide more "algorithm-friendly" information, such as gradients and continuity

(3) Some constraints may be eliminated, which increases the rate of feasible solutions in the newly generated solutions, hence improving the search efficiency

We propose the random proportion and random key (RPK) encoding scheme to represent the location and allocation plan of an express CLAP. RPK brings three advantages over traditional encoding schemes:

(1) RPK dynamically chooses the number of distribution centers in the search process. The solutions with different numbers of distribution centers coexist and evolve in the same swarm.

(2) The allocation of delivery points is determined by the order of elements instead of the value of elements. Then, the candidate capacity is determined by the proportion of the total demand. This mechanism allows the delivery point assignment constraint and capacity/demand constraint to be satisfied simultaneously. There is no need to introduce a punishment term for violation of constraints.

(3) RPK benefits all continuous optimization algorithms, swarm-based or not, by the means of transformation of the search landscape and elimination of constraints.

Data Availability

The experiment results data used to support the findings of this study have been deposited in the Science Data Bank repository (https://www.scidb.cn/s/ziQZba).

Conflicts of Interest

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

Acknowledgments

This research was supported by the National Science Foundation of China (grant no. 51875270) and Philosophy and Social Science Research Project for the Universities of Jiangsu Province (grant no. 2019SJA1912).

References

[1] C. Zhao and B. Zhou, "Impact of express delivery industry's development on transportation sector's carbon emissions: an empirical analysis from China," *Sustainable Times*, vol. 1316 pages, 2021.

[2] State Post Bureau of the People's Republic of China, "Investigation Report on China's Express Delivery Industry 2021-2025," China Express Development Index Report 2021, State Post Bureau of the People's Republic of China, Beijing, China, 2022.

[3] K. Guo and Q. Zhang, "A discrete artificial bee colony algorithm for the reverse logistics location and routing problem," *International Journal of Information Technology and Decision Making*, vol. 16, no. 05, pp. 1339–1357, 2017.

[4] O. Arslan, "The location-or-routing problem," *Transportation Research Part B: Methodological*, vol. 147, pp. 1–21, 2021.

[5] K. Moonsri, K. Sethanan, and K. Worasan, "A novel enhanced differential evolution algorithm for outbound logistics of the poultry industry in Thailand," *Journal of Open Innovation: Technology, Market, and Complexity*, vol. 8, no. 1, p. 15, 2022.

[6] H. Bao, L. Zhou, and L. Liu, "Research on logistics scheduling based on PSO," *AIP Conference Proceedings*, vol. 1864, pp. 1–6, 2017.

[7] N. H. Awad, M. Z. Ali, P. N. Suganthan, and R. G. Reynolds, "An ensemble sinusoidal parameter adaptation incorporated with L-SHADE for solving CEC2014 benchmark problems," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 2958–2965, IEEE, Vancouver, BC, Canada, July 2016.

[8] N. Alghanmi, R. Alotaibi, S. Alshammari, A. Alhothali, O. Bamasag, and K. Faisal, "A survey of location-allocation of points of dispensing during public health emergencies," *Frontiers in Public Health*, vol. 10, 2022.

[9] S. Chandra, M. Sarkhel, and A. K. Vatsa, "Capacitated facility location–allocation problem for wastewater treatment in an industrial cluster," *Computers & Operations Research*, vol. 132, no. April, Article ID 105338, 2021.

[10] H. Tupsamudre, S. Saurabh, A. Ramamurthy, M. Gharote, and S. Lodha, "A divide and conquer approach for web services

location allocation problem," *Association for Computing Machinery*, vol. 1, no. 1, 2021.

[11] S. Shiripour, M. Amiri-Aref, and I. Mahdavi, "The capacitated location-allocation problem in the presence of k connections," *Applied Mathematics*, vol. 02, no. 08, pp. 947–952, 2011.

[12] T. Y. Pham, H. M. Ma, and G. T. Yeo, "Application of fuzzy delphi TOPSIS to locate logistics centers in vietnam: the logisticians' perspective," *The Asian Journal of Shipping and Logistics*, vol. 33, no. 4, pp. 211–219, 2017.

[13] L. Yang, X. Ji, Z. Gao, and K. Li, "Logistics distribution centers location problem and algorithm under fuzzy environment," *Journal of Computational and Applied Mathematics*, vol. 208, no. 2, pp. 303–315, 2007.

[14] I. Karaoglan, F. Altiparmak, I. Kara, and B. Dengiz, "The location-routing problem with simultaneous pickup and delivery: formulations and a heuristic approach," *Omega*, vol. 40, no. 4, pp. 465–477, 2012.

[15] A. Vafaeinejad, S. Bolouri, A. A. Alesheikh, M. Panahi, and C. W. Lee, "The capacitated location-allocation problem using the vaomp (Vector assignment ordered median problem) unified approach in gis (geospatial information systam)," *Applied Sciences*, vol. 10, no. 23, pp. 8505–8522, 2020.

[16] C. Zheng, X. Zhao, and J. Shen, "Research on location optimization of metro-based underground logistics system with Voronoi diagram," *IEEE Access*, vol. 8, pp. 34407–34417, 2020.

[17] W. Tu, Z. Fang, Q. Li, S. L. Shaw, and B. Y. Chen, "A bi-level Voronoi diagram-based metaheuristic for a large-scale multi-depot vehicle routing problem," *Transportation Research Part E: Logistics and Transportation Review*, vol. 61, pp. 84–97, 2014.

[18] J. Zhou and B. Liu, "Modeling capacitated location-allocation problem with fuzzy demands," *Computers & Industrial Engineering*, vol. 53, no. 3, pp. 454–468, 2007.

[19] S. M. Mousavi and S. T. A. Niaki, "Capacitated location allocation problem with stochastic location and fuzzy demand: a hybrid algorithm," *Applied Mathematical Modelling*, vol. 37, no. 7, pp. 5109–5119, 2013.

[20] F. Meng, Q. Ji, H. Zheng, H. Wang, and D. Chu, "Modeling and solution algorithm for optimization integration of express terminal nodes with a joint distribution mode," *Journal of Organizational and End User Computing*, vol. 33, no. 4, pp. 142–166, 2021.

[21] X. P. Xu, X. T. Shi, and F. Wang, "Solving logistics distribution center location problem using a wolf pack algorithm," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 20, no. 6–7, pp. 1269–1273, 2017.

[22] B. Sahin, H. Yilmaz, Y. Ust, A. F. Guneri, and B. Gulsun, "An approach for analysing transportation costs and a case study," *European Journal of Operational Research*, vol. 193, no. 1, pp. 1–11, 2009.

[23] M. Greer, "The theory of natural monopoly and literature review," *Electricity Marginal Cost Pricing*, vol. 42, pp. 15–38, 2012.

[24] F. Peñuñuri, C. Cab, O. Carvente, M. A. Zambrano-Arjona, and J. A. Tapia, "A study of the Classical Differential Evolution control parameters," *Swarm and Evolutionary Computation*, vol. 26, pp. 86–96, 2016.

[25] C. A. Chen and T. C. Chiang, "Adaptive differential evolution: a visual comparison," in *Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC)*, pp. 401–408, IEEE, Sendai, Japan, May 2015.

[26] J. Zhang and A. C. Sanderson, "JADE: self-adaptive differential evolution with fast and reliable convergence performance," in *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, pp. 2251–2258, IEEE, Singapore, September 2007.

[27] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for Differential Evolution," in *Proceedings of the 2013 IEEE Congress on Evolutionary Computation*, no. 3, pp. 71–78, IEEE, Cancun, Mexico, June 2013.

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

[29] N. H. Awad, M. Z. Ali, and P. N. Suganthan, "Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems," in *Proceedings of the 2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings*, pp. 372–379, IEEE, Donostia, Spain, June 2017.

[30] Google Maps, "Google Inc," 2022, https://www.google.com/maps.

Hindawi
Discrete Dynamics in Nature and Society
Volume 2022, Article ID 5129098, 12 pages
https://doi.org/10.1155/2022/5129098

Research Article

Multistrategy Harris Hawks Optimization Algorithm Using Chaotic Method, Cauchy Mutation, and Elite Individual Guidance

Lei Wen,[1] **Guopeng Wang**,[2] **Longwang Yue**,[3] **Xiaodan Liang**,[1] **and Hanning Chen**[1]

[1]*School of Computer Science and Technology, Tiangong University, Tianjin, China*
[2]*The Open University of China, Research and Development Center for E-Learning, Ministry of Education, Beijing, China*
[3]*School of Mechanical and Electrical Engineering, Henan University of Technology, Zhengzhou, Henan, China*

Correspondence should be addressed to Guopeng Wang; wangguopeng@sohu.com and Longwang Yue; gfh110058@163.com

Received 14 June 2022; Revised 13 July 2022; Accepted 5 August 2022; Published 28 August 2022

Academic Editor: Shi Cheng

Aiming at the shortcomings of the Harris hawks optimization algorithm (HHO), such as poor initial population diversity, slow convergence speed, poor local optimization ability, and easily falling into local optimum, a Harris hawks optimization algorithm (CCCHHO) integrating multiple mechanisms is proposed. First, the population diversity is enhanced by the initialization of the chaotic method. Second, the cosine function is used to better simulate the characteristics of the periodic change of the energy of the prey in the repeated contests with the group of hawks, to better balance the exploration and exploitation of the algorithm. Third, Cauchy mutation on the optimal individual in the exploration phase is performed, and the characteristics of the Cauchy distribution to enhance the diversity of the population are used, which can effectively prevent the algorithm from falling into the local optimum. Fourth, the local optimization ability of the algorithm by using the ergodicity of the chaotic system in the exploitation phase to perform a chaotic local search for the optimal individual is enhanced, which can effectively jump out after the algorithm falls into the local optimum. Finally, we use the elite individuals of the population to guide the position update of the population's individuals, fully communicate with the dominant individuals, and speed up the convergence speed of the algorithm. Through the simulation experiments on CCCHHO with 11 different benchmark functions, CCCHHO is better than the gray wolf optimization algorithm (GWO), the Salp swarm algorithm (SSA), the ant lion optimization algorithm (ALO), and three improved HHO algorithms in terms of convergence speed and optimization accuracy, whether it is a unimodal benchmark function or a multimodal benchmark function. The experimental results show that CCCHHO has excellent algorithm efficiency and robustness.

1. Introduction

In recent years, meta-heuristic algorithms have attracted more and more scholars' attention. Because meta-heuristic algorithms are simple to implement, have no layer dependencies, and can jump out of local optima, they are used to solve different problems [1–4].

Meta-heuristic algorithms can be divided into four categories: physics-based, evolution-based, population-based, and human-based. The physics-based aspect mainly simulates the physical rules in the universe. Common physics-based meta-heuristic algorithms include gravitational search algorithm (GSA) [5], central force optimization (CFO) [6], and Black Hole optimization algorithm (BH) [7]. The evolution-based methods are inspired by biological evolution and classical evolutionary algorithms such as genetic algorithm (GA) [8], differential evolution algorithm (DE) [9], and biogeography-based optimizer(BBO) [10]. The population-based algorithms are inspired by the collective behavior of biological populations and classical swarm intelligence algorithms such as ant colony optimization(ACO) [11], particle swarm optimization algorithm (PSO) [12],

artificial bee colony optimization algorithm (ABC) [13], and Monarch butterfly optimization algorithm [14]. More novel population-based algorithms are gray wolf optimization algorithm (GWO) [15], wolf pack optimization algorithm (WPA) [16], dragonfly algorithm (DA) [17], whale optimization algorithm (WOA) [18], ant lion optimization algorithm (ALO) [19], and Salp swarm algorithm (SSA) [20]. The human-based algorithms are inspired by human behavior, such as brain storm optimization (BSO) [21].

Harris hawks optimization algorithm (HHO) [22]is a swarm intelligence optimization algorithm proposed in recent years, which is derived from the group hunting behavior of Harris hawks. Because HHO has a simple structure, easy implementation, and high performance, it has attracted the attention of a large number of researchers since the algorithm was proposed. The researchers have improved the basic HHO algorithm in different aspects, and some of the improved algorithms have been applied to different fields. ElSayed et al. [23] used HHO combined with sequential quadratic programming (SQP) to the optimal coordination problem of directional overcurrent relays incorporating distributed generation. Abbasi et al. [24]used the chaos method, Gaussian mutation, differential evolution, and other methods to improve the basic HHO and apply the improved algorithm to the fatigue life analysis of tapered roller bearings. Jouhari et al. [25]introduced the Salp swarm algorithm (SSA) into the basic HHO, and the position update mechanism is selected through a parameter, and the improved algorithm is applied to the scheduling problem. Chen et al. [26]introduced chaotic drift mechanism into the basic HHO to improve the algorithm, and the improved algorithm is applied to the parameter identification problem of photovoltaic cells and modules. Jia et al. [27]used dynamic parameters to adjust prey escape energy factor and mutation mechanism to improve the basic HHO, and used the improved algorithm for the satellite image segmentation problem. Qu et al. [28] enhanced the information exchange between population individuals and introduced escape energy factors with chaotic disturbances to improve the basic HHO algorithm.

Although different scholars have proposed different improved HHO algorithms, they are not suitable for all optimization problems. Based on this, it makes sense to develop more efficient and accurate algorithms. In this article, a CCCHHO algorithm that integrates multiple strategies is proposed. The algorithm initializes the population through the chaotic method and uses the ergodicity of the chaotic system to enhance the diversity of the population, and uses the characteristics of the cosine function to periodically trigger the update of the prey escape energy that more efficiently balances the exploration and exploitation of the algorithm, and introduces a mutation strategy to enhance the global exploration ability that can prevent the algorithm from falling into the local optimum. The use of chaotic local search can effectively jump out after the algorithm falls into the local optimum and improve the local optimization ability of the algorithm at the same time. Finally, the elite individual guidance mechanism is used to update the population's individual position and use the greedy mechanism to obtain the optimal population as the initial population of the next iteration, which enhances the diversity of the population to accelerate the convergence speed of the algorithm. In order to verify the performance of the proposed algorithm, 11 benchmark test functions are used for simulation analysis and compared with other meta-heuristic algorithms and some improved HHO algorithms.

The structure of this article is as follows. Section 2 introduces the basic HHO algorithm. Section 3 introduces the proposed CCCHHO algorithm. The fourth section is a simulation experiment, and the results are analyzed at the same time. Section 5 expounds on the conclusions and future work.

2. Harris Hawks Optimization Algorithm (HHO)

The Harris hawks optimization algorithm is a meta-heuristic algorithm inspired by the Harris hawks hunting. HHO contains multiple search mechanisms depending on the different strategies adopted by the hawk at different phases. A detailed description of these search mechanisms is given below.

2.1. Exploration Phase. At this phase, the Harris hawks update position with two different strategies based on the probability q. The formulas are as follows:

$$X(t+1)=\begin{cases} X_{\text{rand}}(t)-r_1\left|X_{\text{rand}}(t)-2r_2X(t)\right|, & q\geq 0.5,\\ X_{\text{rabbit}}(t)-X_m(t)-r_3\left(\text{LB}+r_4(\text{UB}-\text{LB})\right), & q<0.5,\end{cases} \tag{1}$$

$$X_m(t)=\frac{1}{N}\sum_{i=1}^{N}X_i(t), \tag{2}$$

where $X(t+1)$ and $X(t)$ represent the position vector of hawks in iteration $t+1$ and t, respectively, $X_{\text{rabbit}}(t)$ is the position of rabbit, $X_m(t)$ is the average position of the current population of hawks, $X_{\text{rand}}(t)$ represents randomly selected individuals in the current population, r_1, r_2, r_3, r_4, and q are random numbers between 0 and 1, and LB and UB are the upper and lower bounds of the population, respectively, and N is the population number.

2.2. Transition from Exploration to Exploitation. HHO realizes the transformation of exploration and exploitation through the escape energy factor E. The formula as follows:

$$E=2E_0\left(1-\frac{t}{T}\right), \tag{3}$$

where E_0 is a random number between -1 and 1, T is the maximum number of iterations, and t is the current number of iterations. When $|E|\geq 1$, HHO performs the global search. Otherwise, local exploitation is performed.

2.3. Exploitation Phase. During the exploitation phase, HHO uses four different strategies to update location and decides which strategy to use by the escape energy factor E and the random number $r\in(0,1)$, and r indicates the

chance of the prey to escape before the surprise pounce. When $r \geq 0.5$, it cannot escape. When $r < 0.5$, it can.

2.4. Soft Besiege. When $r \geq 0.5$ and $|E| \geq 0.5$, the prey has enough energy to try to escape by jumping, but ultimately, it cannot. The hawk's position update formula is as follows:

$$X(t+1) = X_{\text{rabbit}}(t) - X(t) - E\left|\text{JX}_{\text{rabbit}}(t) - X(t)\right|, \quad (4)$$

where $J = 2(1 - r_5)$ represents the random jump intensity when the prey escapes, and r_5 is a random number between 0 and 1.

2.5. Hard Besiege. When $r \geq 0.5$ and $|E| < 0.5$, the prey is captured by the hawks with lower energy. The hawk's position update formula is as follows:

$$X(t+1) = X_{\text{rabbit}}(t) - E\left|X_{\text{rabbit}}(t) - X(t)\right|. \quad (5)$$

2.6. Soft Besiege with Progressive Rapid Dives. When $r < 0.5$ and $|E| \geq 0.5$, the prey has enough energy to ensure a successful escape, so the soft besiege strategy of fast dive is implemented, and the position update formulas are as follows:

$$Y = X_{\text{rabbit}}(t) - E\left|\text{JX}_{\text{rabbit}}(t) - X(t)\right|, \quad (6)$$

$$Z = Y + S \times \text{LF}(D), \quad (7)$$

$$X(t+1) = \begin{cases} Y, f(Y) < f(X(t)), \\ Z, f(Z) < f(X(t)), \end{cases} \quad (8)$$

where D is the problem dimension, S is the D dimension random row vector, and LF is the Levf function. The formula is as follows:

$$\text{LF}(x) = 0.01 \times \frac{u \times \sigma}{|v|^{1/\beta}}, \qquad \sigma = \left(\frac{\Gamma(1+\beta) \times \sin(\pi\beta/2)}{\Gamma(1+\beta/2) \times \beta \times 2^{(\beta-1/2)}}\right)^{1/\beta}, \quad (9)$$

where u and v are random numbers between 0 and 1, and β is 1.5.

2.7. Hard Besiege with Progressive Rapid Dives. When $r < 0.5$ and $|E| < 0.5$, prey energy is lower, but escape is still possible. A hard besiege strategy of rapid dive is implemented to reduce the average distance from the prey, and the position update formulas are as follows:

$$X(t+1) = \begin{cases} Y, & f(Y) < f(X(t)), \\ Z, & f(Z) < f(X(t)), \end{cases} \quad (10)$$

$$Y = X_{\text{rabbit}}(t) - E\left|\text{JX}_{\text{rabbit}}(t) - X_m(t)\right|, \quad (11)$$

$$Z = Y + S \times \text{LF}(D), \quad (12)$$

where D is the problem dimension, S is the D dimension random row vector, LF is the Levf function (Equation (9)), $X_{\text{rabbit}}(t)$ is the position of rabbit, and $X_m(t)$ is the average position of the current population of hawks.

3. Harris Hawks Optimization Algorithm with Multiple Strategies (CCCHHO)

Aiming at the problems existing in the HHO algorithm, the article improves the HHO through various aspects. By introducing the method of Logistic map to replace the random initialization method of the basic HHO algorithm, the characteristics of the chaotic system are used to help the algorithm generate more diverse populations. In the exploration phase, the Cauchy mutation strategy is used to enhance the global search ability that helps the algorithm jump out of the local optimum. A new formula is used to update the energy factor E to better balance the global search and local exploitation. In the exploitation phase, the chaotic local search strategy is used to enhance the local search ability. Finally, introduce the elite guidance strategy, which uses the dominant group to guide the update of the population individuals, and then, use the greedy strategy to save the better individuals to speed up the convergence speed of the algorithm.

3.1. Chaotic Maps. Chaotic systems have the characteristics of randomness and ergodicity. More diverse populations can be generated by using these characteristics, thereby improving the performance and speeding up the convergence speed of the algorithm. Kaur et al. [29] used the chaotic map, which generates the initialization population instead of the randomly generated population, which makes the improved algorithm easy to jump out of the local optimum and improves the optimization performance of the whale algorithm. At present, there are many different chaotic maps in the optimization field [30], mainly including Logistic map, Tent map, and Gauss map. This article uses the Logistic map to generate the initialization population. The Logistic map is defined as follows:

$$x_{i+1} = \mu x_i(1 - x_i), \quad i = 1, 2, \&, N-1, \quad (13)$$

where μ is 4, x_1 is a random number between 0 and 1, and N is the number of population's individuals.

The initial population position generated by using the Logistic map is more uniform distribution of population position compared to the randomly generated, which increases the diversity of the population and expands the search range of the hawks in space. To a certain extent, it improves the shortcomings of HHO that it is easy to fall into local optimum.

3.2. Cauchy Mutation. Cauchy mutation originates from the Cauchy distribution, and the standard Cauchy distribution probability density is as follows:

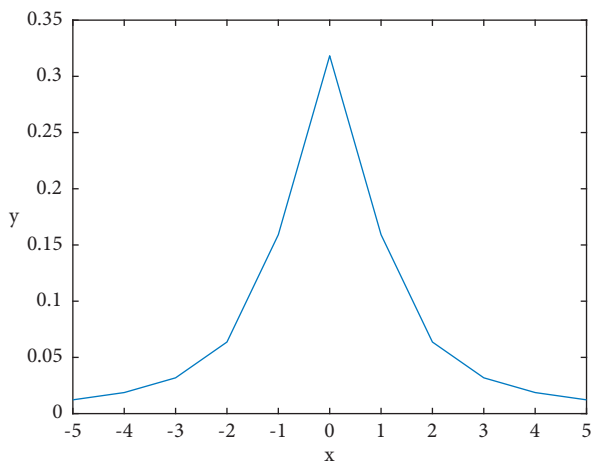


Figure 1: Standard Cauchy distribution probability density function curve.

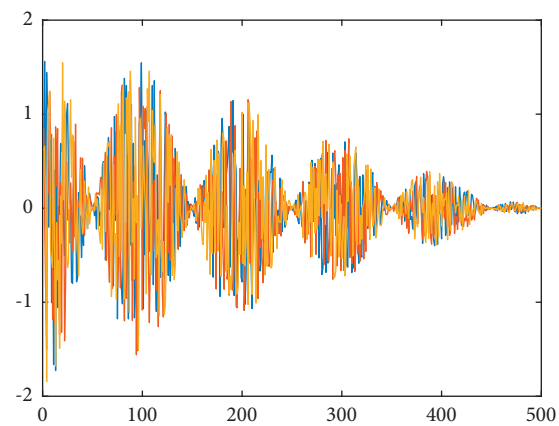


Figure 2: Energy escape factor.

$$f(x) = \frac{1}{\pi} \cdot \frac{1}{1 + x^2}, \quad x \in (-\infty, +\infty). \tag{14}$$

Figure 1 shows that the probability distribution of the Cauchy distribution function is closer to the horizontal axis in the horizontal direction, and the slower it changes. Therefore, the Cauchy distribution can be regarded as infinite. So in terms of probabilities, the Cauchy distribution has a wider distribution range [31]. This means that by using the random numbers generated by the Cauchy distribution as the perturbation factor in the optimization process, one can obtain a relatively broad search space, which can prevent the algorithm from falling into the local optimum in the exploration phase in a way, and after falling into the local optimum, it is easier to jump out the optimum. After obtaining the optimal solution of the current population, use the following formula to update the current global optimal solution:

$$X'_{\text{best}} = X_{\text{best}} + X_{\text{best}} \times \text{cauchy}(0, 1) \frac{1}{2}. \tag{15}$$

Here, cauchy$(0, 1)$ is the standard Cauchy distribution.

3.3. Nonlinear Energy Factor E Based on Cosine Strategy. In the HHO algorithm, the energy factor E is an important parameter to balance the exploration phase and the exploitation phase. The larger $|E|$, the more inclined the HHO algorithm is to perform the exploration. Conversely, the more inclined it is to perform the exploitation. In the HHO algorithm, the energy factor E decreases linearly from large to small, which cannot effectively describe the real situation of Harris hawks rounding up prey in nature. In the multiround game of the Harris hawks and the prey, the energy of the prey cannot be simply reflected by linear changes. The energy of the prey should change periodically and eventually reach zero to be captured by the hawks. During each round of rounding up, the prey will get a short rest to recover a small amount of energy, but over time, the energy recovered by the prey will gradually decrease until the energy reaches zero. In this article, the cosine function is used to describe the periodic change of prey energy. The formula is as follows:

$$E = 2 \times \cos\left(\frac{5\pi t}{T}\right) \times \left(1 - \frac{t}{T}\right) \times (2\text{rand} - 1). \tag{16}$$

Figure 2 shows the change of energy escape factor during iteration. As we can see, the energy escape factor changes periodically until it becomes zero, which well describes the energy change of prey in the process of being rounded up.

3.4. Chaotic Local Search. Local search is one of the effective methods to prevent the algorithm from falling into the local optimum. In most cases, the solution is near the local optimum, and the algorithm cannot obtain it. After the algorithm falls into the local optimum, by using the local search method to search for the vicinity of the local optimal solution, one can effectively jump out of the local optimum and improve the performance of the algorithm. However, sometimes, the local method does not produce ideal results. The ergodic characteristics of chaotic systems can be considered, and the system starts from an initial state and follows its own motion law and experiences all state points in its attraction space without repetition for a long enough time. If a chaotic sequence of length k is superimposed on the optimal individual of the current population, it is equivalent to carrying out k times local search without repetition in the neighbourhood of the optimal individual of the current population. Introduce the chaotic local search strategy [32], and the ergodicity of the chaotic system can effectively prevent the algorithm from falling into the local area, and at the same time, the search efficiency and search range are enhanced compared with local search. The pseudocode of the chaotic local search algorithm is given in Algorithm 1.

In this article, Equation (13) is used in the chaotic search algorithm that the Logistic map produces a chaotic sequence. The chaotic sequence value generated by the Logistic map is between 0 and 1. If the value is directly superimposed on any dimension of the optimal

```
Initialization chaotic search times D
Generate a chaotic sequence of length D by using a chaotic map
Get the best individual in the current population X_best
Setting up the chaotic search counter k = 0
While k < D
  Superimpose an item of the chaotic sequence on any dimension in X_best to form a new individual X_best^cls
  Calculate the fitness value f(X_best^cls) of new individual X_best^cls
  if f(X_best^cls) < f(X_best)
  Update the optimal individual and fitness value of the current population
  end if
  k = k + 1
end while
```

Algorithm 1: Chaotic local search.

```
Initialize population size N, number of iterations T
Use Equation (13) to perform chaotic strategy initialization population X_i (i = 1, 2, ..., N)
Set the current number of iterations t = 0
while t < T
  Calculate the fitness value of hawks
  Set the best position as the position of the prey X_rabbit
  for each individual X_i
    Update escape energy E and jump strength J by using Equation (16)
    if(|E| ≥ 1)
      Update location by using Equation (1)
      Get the optimum of the current population and use Equation (15) to carry out the Cauchy mutation
    end if
    if(|E| < 1)
      if(r ≥ 0.5, |E| ≥ 0.5)
        Update location by using Equation (4)
      else if(r ≥ 0.5, |E| < 0.5)
        Update location by using Equation (5)
      else if(r ≥ 0.5, |E| < 0.5)
        Update location by using Equation (8)
      else if(r ≥ 0.5, |E| < 0.5)
        Update location by using Equation (10)
      end if
    Carry out chaotic local search
    end if
  end for
  Carry out elite individual guidance
  t = t + 1
end while
```

Algorithm 2: CCCHHO algorithm.

individual, the search will only be carried out in one direction, so that the local search performance will be greatly reduced. To solve this problem, this article uses the number that chaotic search counter k to periodically adjust the search direction. The position update formula is as follows:

Table 1: Description of benchmark functions.

Function	V_no	Range	$f_{\min}$
$F_1(x) = \sum_{i=1}^{n} x_i^2$	30	[−100, 100]	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[−10, 10]	0
$F_3(x) = \max\{\lvert x_i \rvert, 1 \le i \le n\}$	30	[−100, 100]	0
$F_4(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[−30, 30]	0
$F_5(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	[−100, 100]	0
$F_6(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	[−500, 500]	-418.9829×5
$F_7(x) = -20\exp(-0.2\sqrt{(1/n)\sum_{i=1}^{n} x_i^2}) - \exp((1/n)\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + e$	30	[−32, 32]	0
$F_8(x) = (\pi/n)\{10\sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + (x_i + 1/4)u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m x_i > a, \\ 0 - a < x_i < a, \\ k(-x_i - a)^m x_i < -a, \end{cases}$	30	[−50, 50]	0
$F_9(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	[−50, 50]	0
$F_{10}(x) = ((1/500) + \sum_{j=1}^{25} (1/j + \sum_{i=1}^{2} (x_i - a_{ij})^6))^{-1}$	2	[−65, 65]	1
$F_{11}(x) = \sum_{i=1}^{11} [a_i - (x_1(b_i^2 + b_i x_2)/b_i^2 + b_i x_3 + x_4)]^2$	4	[−5, 5]	0.00030

$$X_{\text{best}}^{\text{cls}} = X_{\text{best}} + (-1)^k \beta_k, \tag{17}$$

where X_{best} is the optimal individual in the current population, $X_{\text{best}}^{\text{cls}}$ is the new individual after a chaotic local search, k is the number that chaotic search counter, and β_k is the chaotic sequence value generated by the Logistic map.

3.5. Elite Individual Guidance Mechanism. In the gray wolf optimization algorithm, it is proposed that the position of the wolf pack follows the guidance of three individuals in the group, α wolf, β wolf, and δ wolf, which can effectively improve the search efficiency. Based on this, this article introduces an elite individual guidance mechanism. The top three better individuals in the current population are selected as leaders to guide the location update of other individuals, so that the information exchange between the population individuals and the better individual in the population is strengthened, thereby enhancing the diversity of the population. The hawk's position update formulas are as follows:

$$X' = X + A * \left(2X_\alpha - X_\beta - X_\delta\right), \tag{18}$$

$$A = 2ar - a, \tag{19}$$

where X is the current individual position, X' is the updated position, X_α is the optimal solution in the current population, X_β is the suboptimal solution in the current population, X_δ is the third optimal solution in the current population, a is linearly decreased from 2 to 0 in the iterative process, and r is a random number between 0 and 1.

Based on the greedy mechanism, the better one is selected as a new individual, and the rabbit position and rabbit energy are updated at the same time.

3.6. The Pseudocode of CCCHHO. Algorithm 2 pseudocode describes the details of CCCHHO.

4. Results and Discussion

In order to verify the performance of the proposed CCCHHO algorithm, this article tests the algorithm through 11 benchmark functions and gives the experimental results and analysis. The test functions can be divided into two groups: F1~ F5 are unimodal benchmark functions, in which the characteristics with the unique global optimal value can evaluate the local exploitation capabilities of different optimization algorithms, and F6~ F11 are multimodal benchmark functions, which can evaluate the global exploration of different algorithms and the ability to avoid local optimum. According to the best, mean, worst, and standard deviation of the results, the proposed CCCHHO is compared with other optimization algorithms, such as GWO, SSA, and ALO. At the same time, it is compared with the other improved HHO algorithms. The information of the benchmark functions F1–F11 is shown in Table 1.

4.1. Experimental Settings. All algorithms run using MATLAB R2019b on a computer with 16G memory, AMD4800U. The population size of all algorithms is set to 30, the maximum number of iterations is set to 500, and each algorithm loop executes 30 times.

Table 2: The result of the comparison of CCCHHO with HHO, GWO, SSA, and ALO.

Function	Index	CCCHHO	HHO	GWO	SSA	ALO
F1	**Best**	**0.00E + 00**	7.44E − 115	6.94E − 30	3.42E − 08	2.25E − 09
	Avg	**0.00E + 00**	1.42E − 98	9.05E − 28	6.22E − 07	9.83E − 09
	Worst	**0.00E + 00**	2.90E − 97	5.04E − 27	7.13E − 06	5.86E − 08
	Std	**0.00E + 00**	5.43E − 98	1.33E − 27	1.50E − 06	1.00E − 08
F2	**Best**	**6.30E − 238**	2.38E − 60	1.93E − 17	5.10E − 06	1.68E − 05
	Avg	**1.69E − 180**	5.47E − 52	8.89E − 17	2.09E − 03	5.72E − 01
	Worst	**5.07E − 179**	4.53E − 51	1.99E − 16	5.18E − 02	3.95E + 00
	Std	**0.00E + 00**	1.22E − 51	5.33E − 17	9.47E − 03	9.83E − 01
F3	**Best**	**3.04E − 239**	2.83E − 57	9.35E − 08	9.21E − 06	2.04E − 04
	Avg	**3.72E − 209**	3.16E − 50	6.08E − 07	2.07E − 05	4.69E − 03
	Worst	**1.07E − 207**	1.73E − 49	1.88E − 06	3.48E − 05	5.53E − 02
	Std	**0.00E + 00**	5.37E − 50	5.15E − 07	6.34E − 06	1.16E − 02
F4	**Best**	**0.00E + 00**	3.34E − 04	2.57E + 01	3.84E + 00	5.57E − 04
	Avg	**0.00E + 00**	1.15E − 02	2.69E + 01	1.34E + 02	1.07E + 02
	Worst	**0.00E + 00**	6.87E − 02	2.87E + 01	2.03E + 03	6.67E + 02
	Std	**0.00E + 00**	1.61E − 02	8.70E − 01	3.79E + 02	1.82E + 02
F5	**Best**	**0.00E + 00**	2.60E − 10	5.27E − 05	4.30E − 10	1.83E − 09
	Avg	**0.00E + 00**	1.01E − 04	7.06E − 01	9.71E − 10	1.11E − 08
	Worst	**0.00E + 00**	3.43E − 04	1.43E + 00	1.98E − 09	8.91E − 08
	Std	**0.00E + 00**	1.08E − 04	3.59E − 01	4.14E − 10	1.57E − 08
F6	**Best**	**−1.26E + 04**	**−1.26E + 04**	−7.79E + 03	−3.71E + 03	−3.56E + 03
	Avg	**−1.26E + 04**	**−1.26E + 04**	−6.02E + 03	−2.83E + 03	−2.54E + 03
	Worst	**−1.26E + 04**	−1.23E + 04	−3.77E + 03	−2.25E + 03	−1.81E + 03
	Std	**1.85E − 12**	5.55E + 01	7.57E + 02	3.44E + 02	5.44E + 02
F7	**Best**	**8.88E − 16**	**8.88E − 16**	7.19E − 14	8.15E − 06	2.33E − 05
	Avg	**8.88E − 16**	**8.88E − 16**	1.05E − 13	7.30E − 01	3.82E − 01
	Worst	**8.88E − 16**	**8.88E − 16**	1.36E − 13	2.32E + 00	2.01E + 00
	Std	**0.00E + 00**	**0.00E + 00**	1.65E − 14	9.43E − 01	6.73E − 01
F8	**Best**	**1.57E − 32**	3.91E − 08	1.28E − 02	1.33E − 11	9.98E − 10
	Avg	**1.57E − 32**	5.78E − 06	6.31E − 02	9.92E − 01	2.78E + 00
	Worst	**1.57E − 32**	2.89E − 05	5.57E − 01	6.37E + 00	8.15E + 00
	Std	**5.57E − 48**	8.34E − 06	9.56E − 02	1.44E + 00	2.58E + 00
F9	**Best**	**1.35E − 32**	1.28E − 07	2.01E − 01	5.81E − 11	2.15E − 09
	Avg	**1.35E − 32**	1.01E − 04	5.58E − 01	2.56E − 03	2.13E − 03
	Worst	**1.35E − 32**	5.26E − 04	9.24E − 01	1.10E − 02	3.08E − 02
	Std	**5.57E − 48**	1.29E − 04	1.93E − 01	4.73E − 03	6.37E − 03
F10	**Best**	**9.98E − 01**	**9.98E − 01**	**9.98E − 01**	**9.98E − 01**	**9.98E − 01**
	Avg	**9.98E − 01**	1.49E + 00	4.23E + 00	1.13E + 00	2.15E + 00
	Worst	**9.98E − 01**	5.93E + 00	1.27E + 01	1.99E + 00	7.87E + 00
	Std	**2.40E − 16**	9.62E − 01	3.80E + 00	3.44E − 01	1.76E + 00
F11	**Best**	**3.08E − 04**	3.09E − 04	**3.08E − 04**	5.14E − 04	5.06E − 04
	Avg	**3.26E − 04**	4.50E − 04	1.81E − 03	4.19E − 03	4.84E − 03
	Worst	**3.65E − 04**	1.40E − 03	2.04E − 02	2.04E − 02	2.12E − 02
	Std	**1.55E − 05**	3.06E − 04	5.05E − 03	7.37E − 03	8.01E − 03

4.2. Comparison of CCCHHO and Other Optimization Algorithms. This section compares CCCHHO with four optimization algorithms: HHO, GWO, SSA, and ALO. Record the optimal value (best), mean value (avg), worst value (worst), and standard deviation (std) of each algorithm. The results of each algorithm are shown in Table 2.

From Table 2, it can be seen that the performance of the CCCHHO algorithm proposed in this article is greatly improved compared with the performance of other swarm intelligence optimization algorithms. For the F1~ F5 function, the results of the four dimensions of CCCHHO are much higher than the other four algorithms, especially the results of F1, F4, and F5 all reach the minimum value of the function. At the same time, the stability of CCCHHO is also far better than the other four algorithms. For the six functions of F6~ F11, the results of F8 and F9 are better than the other four algorithms. And the results of F7 are the same as the HHO algorithm, better than GWO, SSA, and ALO. The mean value of F6 is the same as that of HHO, but the stability is better. It is also better than GWO, SSA, and ALO. The mean values of F10 and F11 are slightly superior, and the stability is also better.

Table 3: The result of comparison of CCCHHO with the improved algorithm.

Function	Index	CCCHHO	THHO	MHHO	OBLHHO
F1	**Best**	**0.00E + 00**	**0.00E + 00**	1.58E − 141	**0.00E + 00**
	Avg	**0.00E + 00**	6.48E − 108	4.94E − 124	7.20E − 299
	Worst	**0.00E + 00**	1.92E − 106	1.01E − 122	2.16E − 297
	Std	**0.00E + 00**	3.51E − 107	1.92E − 123	**0.00E + 00**
F2	**Best**	**6.30E − 238**	2.74E − 178	1.53E − 72	9.02E − 222
	Avg	**1.69E − 180**	1.01E − 56	8.05E − 63	2.35E − 170
	Worst	**5.07E − 179**	1.54E − 55	1.95E − 61	7.02E − 169
	Std	**0.00E + 00**	3.40E − 56	3.57E − 62	**0.00E + 00**
F3	**Best**	**3.04E − 239**	6.44E − 170	9.30E − 71	1.15E − 212
	Avg	**3.72E − 209**	3.41E − 56	1.42E − 63	1.41E − 159
	Worst	**1.07E − 207**	8.97E − 55	2.95E − 62	4.22E − 158
	Std	**0.00E + 00**	1.64E − 55	5.43E − 63	7.70E − 159
F4	**Best**	**0.00E + 00**	3.92E − 08	9.05E − 06	5.63E − 06
	Avg	**0.00E + 00**	6.65E − 03	5.84E − 03	1.01E − 02
	Worst	**0.00E + 00**	3.87E − 02	3.96E − 02	8.90E − 02
	Std	**0.00E + 00**	9.68E − 03	8.69E − 03	2.01E − 02
F5	**Best**	**0.00E + 00**	5.93E − 09	1.89E − 07	5.03E − 07
	Avg	**0.00E + 00**	1.64E − 04	8.92E − 05	1.76E − 04
	Worst	**0.00E + 00**	1.13E − 03	4.92E − 04	8.03E − 04
	Std	**0.00E + 00**	2.92E − 04	1.06E − 04	2.13E − 04
F6	**Best**	**−1.26E + 04**	**−1.26E + 04**	**−1.26E + 04**	**−1.26E + 04**
	Avg	**−1.26E + 04**	−1.25E + 04	−1.25E + 04	−1.25E + 04
	Worst	**−1.26E + 04**	−9.95E + 03	−1.19E + 04	−9.05E + 03
	Std	**1.85E − 12**	4.78E + 02	1.23E + 02	6.43E + 02
F7	**Best**	**8.88E − 16**	8.88E − 16	8.88E − 16	8.88E − 16
	Avg	**8.88E − 16**	8.88E − 16	8.88E − 16	8.88E − 16
	Worst	**8.88E − 16**	8.88E − 16	8.88E − 16	8.88E − 16
	std	**0.00E + 00**	0.00E + 00	0.00E + 00	0.00E + 00
F8	**Best**	**1.57E − 32**	7.43E − 09	1.20E − 09	1.48E − 08
	Avg	**1.57E − 32**	1.11E − 05	5.14E − 06	6.55E − 06
	Worst	**1.57E − 32**	4.83E − 05	3.97E − 05	6.32E − 05
	Std	**5.57E − 48**	1.23E − 05	7.67E − 06	1.25E − 05
F9	**Best**	**1.35E − 32**	4.52E − 08	9.51E − 08	3.10E − 07
	Avg	**1.35E − 32**	2.40E − 04	6.75E − 05	7.72E − 05
	Worst	**1.35E − 32**	9.31E − 04	2.60E − 04	5.53E − 04
	Std	**5.57E − 48**	2.64E − 04	7.84E − 05	1.21E − 04
F10	**Best**	**9.98E − 01**	**9.98E − 01**	**9.98E − 01**	**9.98E − 01**
	Avg	**9.98E − 01**	1.49E + 00	1.66E + 00	1.41E + 00
	Worst	**9.98E − 01**	5.93E + 00	5.93E + 00	2.98E + 00
	Std	**2.40E − 16**	1.26E + 00	1.50E + 00	6.12E − 01
F11	**Best**	**3.08E − 04**	**3.08E − 04**	**3.08E − 04**	3.10E − 04
	Avg	**3.26E − 04**	3.66E − 04	3.85E − 04	3.47E − 04
	Worst	**3.65E − 04**	4.75E − 04	1.68E − 03	4.23E − 04
	Std	**1.55E − 05**	4.93E − 05	2.48E − 04	3.24E − 05

4.3. Comparison of CCCHHO and Other Improved HHO Algorithms. This section compares CCCHHO with THHO [33], MHHO [34], and OBLHHO [35]. We record the optimal value (best), mean value (avg), worst value (worst), and standard deviation (std) of each algorithm, and the results of each algorithm are shown in Table 3.

As can be seen from Table 3, the results of the CCCHHO algorithm in 7 benchmark functions that F1~F5, F8, and F9 are higher than other improved HHO algorithms in 4 dimensions, among which F1, F4, and F5 have reached the minimum value of the function. The mean values of the three functions F6, F10, and F11 are slightly better than other improved algorithms; the results of F7 are the same as those of HHO and the improved HHO algorithm. At the same time, the CCCHHO algorithm has better advantages in the standard deviation of all functions except F7. The results show that the CCCHHO algorithm outperforms the other improved algorithms in terms of accuracy and stability.

The convergence curves of CCCHHO, HHO, and other improved HHO algorithms in each benchmark function are shown in Figures 3 and 4.

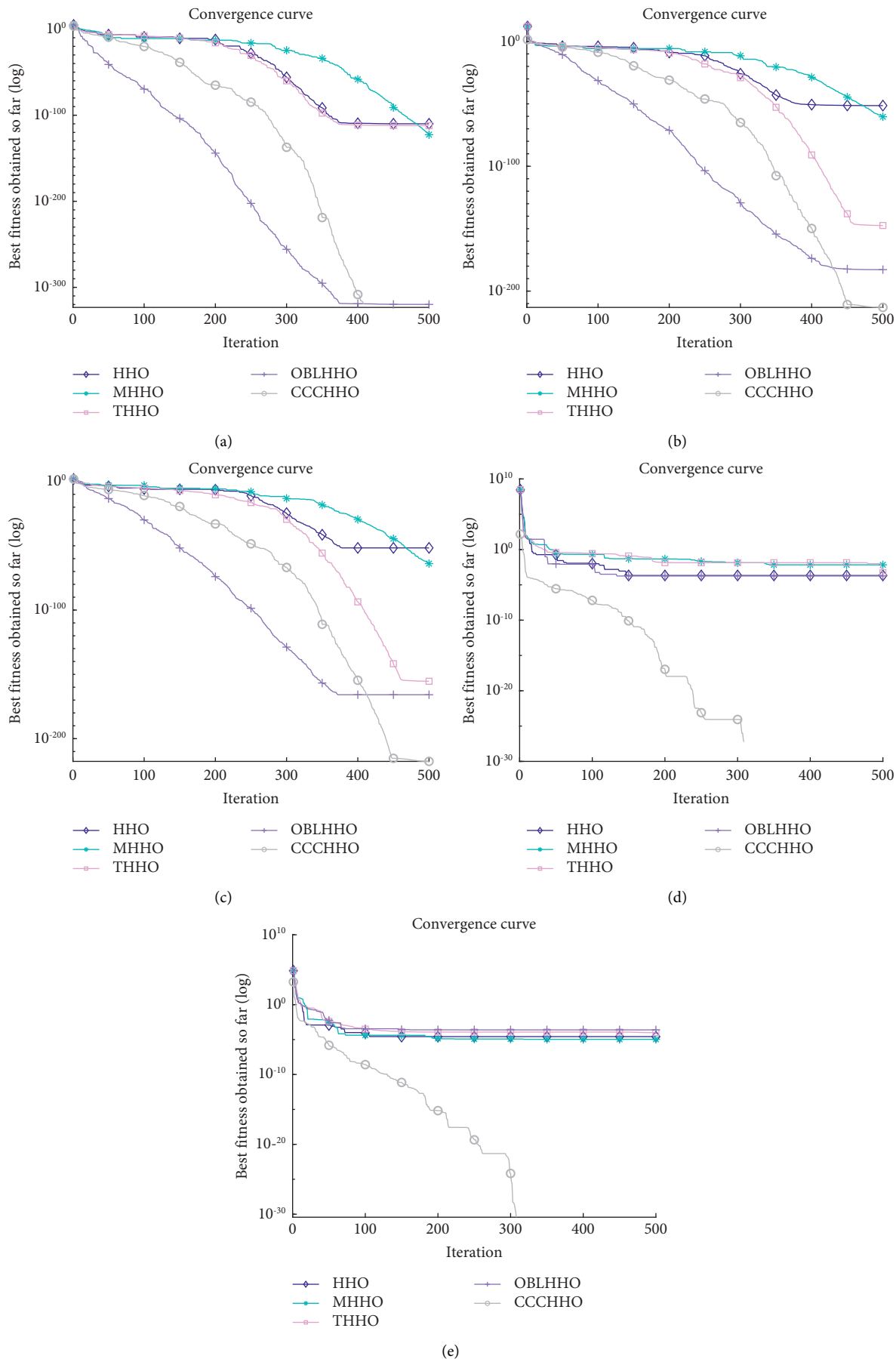


Figure 3: Convergence curve of unimodal function. (a) F1. (b) F2. (c) F3. (d) F4.(e) F5.

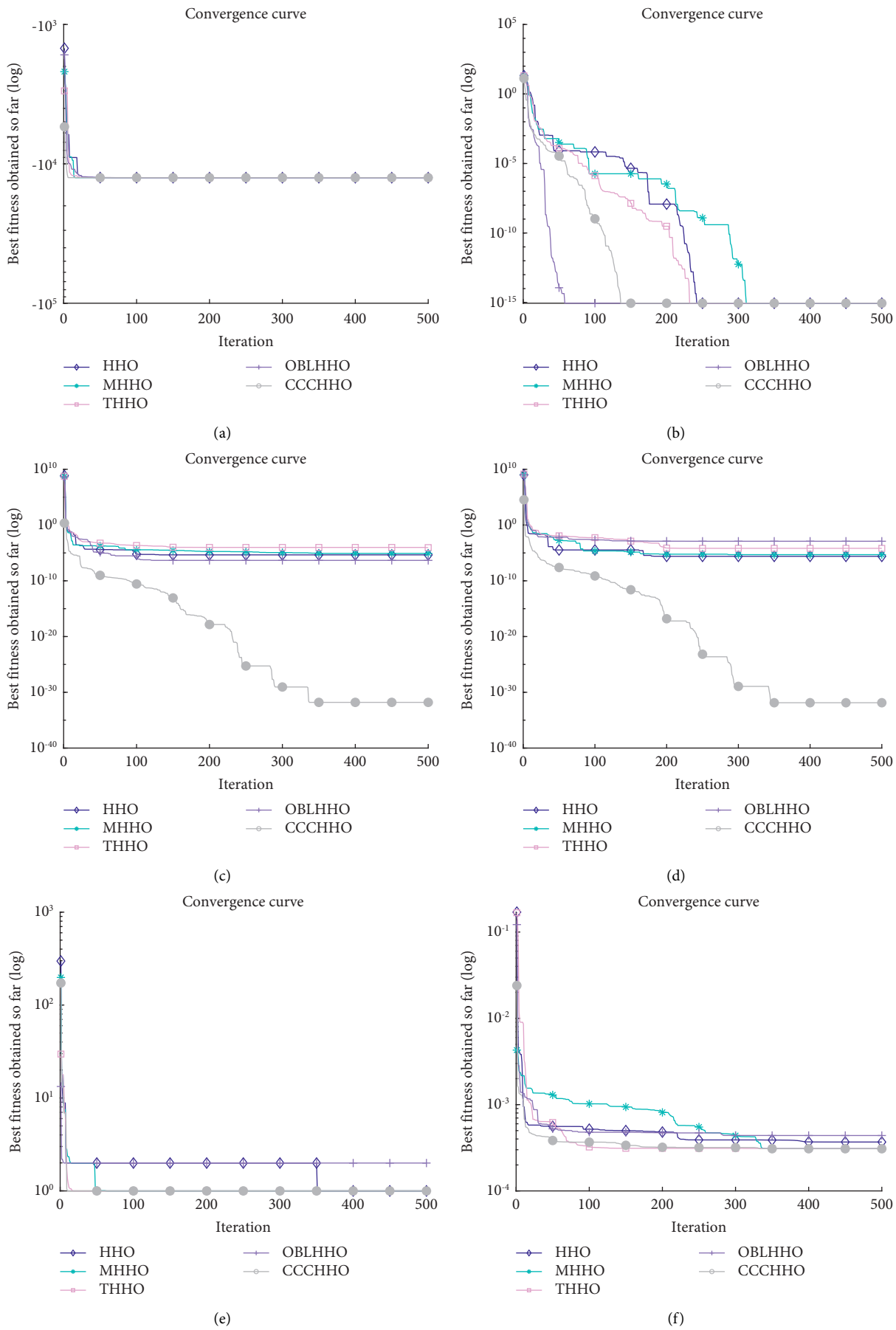


Figure 4: Convergence curve of multimodal function. (a) F6. (b) F7. (c) F8. (d) F9. (e) F10. (f) F11.

Figure 3 shows the convergence curves of each algorithm on the unimodal benchmark function. Among the five improved algorithms, the CCCHHO proposed in this article has the best performance in optimization. In F1~ F3, CCCHHO converges the fastest and has the highest accuracy. In F4 and F5, it can be seen that CCCHHO has fallen into the local optimum many times in the iterative process. In the early phase, it jumped out of the local optimum through the Cauchy mutation strategy, successfully jumped out of the local optimum through the chaotic local search strategy in the later phase of the iteration, and finally found a better solution.

Figure 4 shows the convergence curve of each algorithm on the multimodal benchmark function. For the F6 function, the convergence speed of CCCHHO is relatively fast, and the optimization results are basically the same. In the F7 function, the convergence speed of CCCHHO is only second to OBLHHO. For the F8 function and the F9 function, compared with the other improved HHO algorithms, they fell into local optimum very early and have not jumped out of it. CCCHHO jumped out the local optimal value many times through Cauchy mutation, chaotic local search, and elite individual guidance strategy and finally found a better solution. In F10~ F11, the final results of CCCHHO and the other improved algorithms are better, and the convergence speed of CCCHHO is relatively fast.

5. Conclusions

In this article, a new Harris hawks optimization algorithm with multistrategy (CCCHHO) is proposed by introducing chaotic method initialization population, Cauchy mutation, nonlinear escape energy factor based on cosine function, chaotic local search, and elite individual guidance mechanism to improve the optimization performance of the basic HHO algorithm. In order to verify the performance of the proposed algorithm, 11 different types of benchmark functions were tested to analyze the exploration ability, exploitation ability, and convergence behavior of CCCHHO. The experimental results show that the exploration, exploitation, and convergence speed of CCCHHO is better than that of the basic HHO, three improved HHO algorithms, and three other swarm intelligence optimization algorithms. In future work, higher-dimensional problems will be tested and evaluated, and CCCHHO will be applied to practical engineering problems, such as parameter optimization and shop scheduling.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This work was supported by the National Key R&D Program of China (2019YFB1706302).

References

[1] L. Ma, M. Huang, S. Yang, R. Wang, and X. Wang, "An adaptive localized decision variable analysis approach to large-scale multiobjective and many-objective optimization," *IEEE Transactions on Cybernetics*, vol. 52, no. 7, pp. 6684–6696, 2022.

[2] L. Ma, X. Wang, X. Wang, L. Wang, Y. Shi, and M. Huang, "TCDA: truthful combinatorial double auctions for mobile edge computing in industrial Internet of Things," *IEEE Transactions on Mobile Computing*, p. 1, 2021.

[3] L. Ma, N. Li, Y. Guo et al., "Learning to optimize: reference vector reinforcement learning adaption to constrained many-objective optimization of industrial copper burdening system," *IEEE Transactions on Cybernetics*, pp. 1–14, 2021.

[4] X. Xue, J. Chen, and X. Yao, "Efficient user involvement in semiautomatic ontology matching," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 2, pp. 214–224, 2021.

[5] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: a gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[6] R. A. Formato, "Central force optimization," *Progress In Electromagnetics Research*, vol. 77, no. 1, pp. 425–491, 2007.

[7] A. Hatamlou, "Black hole: a new heuristic optimization approach for data clustering," *Information Sciences*, vol. 222, pp. 175–184, 2013.

[8] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–72, 1992.

[9] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[10] D. Simon, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.

[11] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[12] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the ICNN'95-international conference on neural networks*, pp. 1942–1948, IEEE, Perth, WA, Australia, November 1995.

[13] L. Ma, X. Wang, M. Huang, Z. Lin, L. Tian, and H. Chen, "Two-level master–slave RFID networks planning via hybrid multiobjective artificial bee colony optimizer," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 5, pp. 861–880, 2019.

[14] L. Sun, S. Chen, J. Xu, and Y. Tian, "Improved monarch butterfly optimization algorithm based on opposition-based learning and random local perturbation," *Complexity*, vol. 2019, Article ID 4182148, 2019.

[15] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[16] H.-S. Wu, F. Zhang, and L. Wu, "New swarm intelligence algorithm-wolf pack algorithm," *Systems Engineering and Electronics*, vol. 35, no. 11, pp. 2430–2438, 2013.

[17] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Computing & Applications*, vol. 27, no. 4, pp. 1053–1073, 2016.

[18] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

[19] S. Mirjalili, "The ant lion optimizer," *Advances in Engineering Software*, vol. 83, pp. 80–98, 2015.

[20] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp Swarm Algorithm: a bio-inspired optimizer for engineering design problems," *Advances in Engineering Software*, vol. 114, pp. 163–191, 2017.

[21] L. Ma, S. Cheng, and Y. Shi, "Enhancing learning efficiency of brain storm optimization via orthogonal learning design," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 11, pp. 6723–6742, 2021.

[22] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: algorithm and applications," *Future Generation Computer Systems*, vol. 97, pp. 849–872, 2019.

[23] S. K. ElSayed and E. E. Elattar, "Hybrid Harris hawks optimization with sequential quadratic programming for optimal coordination of directional overcurrent relays incorporating distributed generation," *Alexandria Engineering Journal*, vol. 60, no. 2, pp. 2421–2433, 2021.

[24] A. Abbasi, B. Firouzi, P. Sendur, A. A. Heidari, H. Chen, and R. Tiwari, "Multi-strategy Gaussian Harris hawks optimization for fatigue life of tapered roller bearings," *Engineering with Computers*, pp. 1–27, 2021.

[25] H. Jouhari, D. Lei, M. A. A. Al-qaness et al., "Modified Harris Hawks optimizer for solving machine scheduling problems," *Symmetry*, vol. 12, no. 9, p. 1460, 2020.

[26] H. Chen, S. Jiao, M. Wang, A. A. Heidari, and X. Zhao, "Parameters identification of photovoltaic cells and modules using diversification-enriched Harris hawks optimization with chaotic drifts," *Journal of Cleaner Production*, vol. 244, Article ID 118778, 2020.

[27] H. Jia, C. Lang, D. Oliva, W. Song, and X. Peng, "Dynamic Harris hawks optimization with mutation mechanism for satellite image segmentation," *Remote Sensing*, vol. 11, no. 12, p. 1421, 2019.

[28] C. Qu, W. He, X. Peng, and X. Peng, "Harris hawks optimization with information exchange," *Applied Mathematical Modelling*, vol. 84, pp. 52–75, 2020.

[29] G. Kaur and S. Arora, "Chaotic whale optimization algorithm," *Journal of Computational Design and Engineering*, vol. 5, no. 3, pp. 275–284, 2018.

[30] S. Arora and P. Anand, "Chaotic grasshopper optimization algorithm for global optimization," *Neural Computing & Applications*, vol. 31, no. 8, pp. 4385–4405, 2019.

[31] R. Chunhui, L. Sheng, Z. Weikang, and Z. Weiwei, "Optimization and application of Cauchy mutation camel algorithm," *Computer Engineering and Applications*, vol. 57, no. 21, pp. 87–94, 2021.

[32] Y. Tan, G.-Z. Tan, B. Yang, Z.-C. Zhao, and L. Huang, "Differential evolution algorithm with chaotic-local-search strategy for mixed-integer nonlinear programming problems," *Journal of Chinese Computer Systems*, vol. 33, no. 6, pp. 1306–1309, 2012.

[33] M. A. Al-Betar, M. A. Awadallah, A. A. Heidari, H. Chen, H. Al-Khraisat, and C. Li, "Survival exploration strategies for Harris hawks optimizer," *Expert Systems with Applications*, vol. 168, Article ID 114243, 2021.

[34] Y. Zhang, X. Zhou, and P.-C. Shih, "Modified Harris Hawks optimization algorithm for global optimization problems," *Arabian Journal for Science and Engineering*, vol. 45, no. 12, Article ID 10974, 2020.

[35] O. M. Ismael, O. S. Qasim, and Z. Y. Algamal, "Improving Harris hawks optimization algorithm for hyperparameters estimation and feature selection in v-support vector regression based on opposition-based learning," *Journal of Chemometrics*, vol. 34, no. 11, Article ID e3311, 2020.

Hindawi
Discrete Dynamics in Nature and Society
Volume 2022, Article ID 3259222, 14 pages
https://doi.org/10.1155/2022/3259222

Research Article

Verification of Classification Model and Dendritic Neuron Model Based on Machine Learning

Dongbao Jia,[1,2] **Weixiang Xu**,[1] **Dengzhi Liu**,[1] **Zhongxun Xu**,[1] **Zhaoman Zhong**,[1] **and Xinxin Ban**[3]

[1]*School of Computer Engineering, Jiangsu Ocean University, Lianyungang 222005, China*
[2]*The MOE Key Laboratory of TianQin Project, Sun Yat-Sen University, Zhuhai 519082, China*
[3]*School of Environmental and Chemical Engineering, Jiangsu Ocean University, Lianyungang 222005, China*

Correspondence should be addressed to Zhaoman Zhong; zmzhong@jou.edu.cn and Xinxin Ban; banxx@jou.edu.cn

Received 19 March 2022; Revised 24 May 2022; Accepted 31 May 2022; Published 4 July 2022

Academic Editor: Shi Cheng

Artificial neural networks have achieved a great success in simulating the information processing mechanism and process of neuron supervised learning, such as classification. However, traditional artificial neurons still have many problems such as slow and difficult training. This paper proposes a new dendrite neuron model (DNM), which combines metaheuristic algorithm and dendrite neuron model effectively. Eight learning algorithms including traditional backpropagation, classic evolutionary algorithms such as biogeography-based optimization, particle swarm optimization, genetic algorithm, population-based incremental learning, competitive swarm optimization, differential evolution, and state-of-the-art jSO algorithm are used for training of dendritic neuron model. The optimal combination of user-defined parameters of model has been systemically investigated, and four different datasets involving classification problem are investigated using proposed DNM. Compared with common machine learning methods such as decision tree, support vector machine, *k*-nearest neighbor, and artificial neural networks, dendritic neuron model trained by biogeography-based optimization has significant advantages. It has the characteristics of simple structure and low cost and can be used as a neuron model to solve practical problems with a high precision.

1. Introduction

In the human brain, about tens of billions of interconnected neurons transmit signals through synapses to form a complex neural network to guide human behavior in the network. Neurons are composed of cell bodies with branched dendritic structures, cell membranes, and axons responsible for transmitting nerve signals. The first truly dominant concept of neural network established by scholars has only one neuron unit, known as binary McCulloch-Pitts neuron, which was proposed by McCulloch and Pitts in 1943 [1]. However, it does not consider the nonlinear transmission of cellular signals in dendritic neuron networks. Moreover, as the single-layer McCulloch Pitts neuron model cannot solve the basic nonlinear operation problem [2], it is also criticized as too simple.

Traditional neural networks generally believe that the connection between neurons is very complex, and the brain has strong computing and thinking ability. A single neuron does not need strong computing ability and only needs simple linear summation or nonlinear threshold operation in the process of signal transmission. As a result, the computational potential of individual neurons and their dendrites has been neglected for a long time.

Some researchers have proposed that dendrites perform more complex nonlinear operations in the process of signal transmission, which can improve the computing power of a single neuron [3, 4]. Koch et al. [5] hypothesized that the synaptic interaction at the branch turning point can be realized by Boolean logic operation, which means that the dendritic branch point is responsible for summarizing the current signals from the dendritic branch, its output is the input logic or, and each branch performs logic AND

operation on their synaptic input. However, as small differences in individual neuron morphology can lead to great changes in function, Koch model is difficult to distinguish different synaptic and dendritic morphology to solve specific complex problems. Therefore, the structure of synapses and dendrites requires a plasticity mechanism. Subsequent studies have found the phenomenon of neuronal plasticity, and an important progress in neuronal cell structure has been achieved by proposing neuronal pruning method [6, 7].

Compared with the widely used neural network model under the current mainstream view, the single dendritic neuron network model can deal with more complex nonlinear operations. The dendritic neural network model can carry out more complex nonlinear operation and obtain more accurate results with the same number of neurons. At present, the dendritic neural network model has been applied to many fields and achieved good results, such as live disorders [8], financial time series prediction [9, 10], and breast cancer classification [11–24].

In recent years, the research method of combining metaheuristic algorithm and neural network is more and more widely used. Its basic idea is to use metaheuristic algorithm to continuously adjust the corresponding parameters in neural network, guide the output after obtaining the optimal value, and then replace the obtained parameters into neural network for classification and prediction [25, 26]. The traditional dendritic neural network is optimized by backpropagation algorithm. The backpropagation algorithm is based on the chain derivation rule, which has the problems of falling into local traps, gradient disappearance, and so on [27–29]. This paper proposes a new dendrite neuron model (DNM), which combines metaheuristic algorithm and dendrite neuron model effectively.

Seven different metaheuristic algorithms are compared in the experiment, each of which has its own characteristics. Genetic algorithm (GA) [30–32] is an algorithm of finding the optimal solution based on the simulation of natural selection and genetic mechanism of biological evolution, whose main characteristic is to directly operate on the structure object and free from the restriction of derivation and function continuity. Biogeography-based optimization (BBO) [33–35] has been widely applied to simulate ecological concepts, well-known as its high prevision and strong stability using the representative metaheuristics. Particle swarm optimization (PSO) [36, 37] has been applied to train neural network instead of BP, whose whole searching and updating process follows the current optimal solution. Unlike genetic algorithm, all particles may converge to the optimal solution faster in most cases, and its advantage of evolutionary computation can deal with some problems of nondifferentiable node transfer function or no gradient information. Competitive swarm optimizer (CSO) [38, 39] is a simplified metaheuristic method and, which as a variant of PSO, is not only suitable for multi-point search, but also for local search. On this basis, the competition mechanism is applied, and it is not necessary to update the individual and global optimal value of position. Thus, CSO can balance the local-minimum trapping and convergence rate. Population-based incremental learning (PBIL) [40–42] selects the individuals with the highest fitness in each generation of the group to modify the learning probability and guides the generation of new individuals. Differential evolution (DE) [43, 44] is a stochastic model simulating biological evolution. As with other evolutionary algorithms, DE remains a global search strategy based on population, and for a further step, the process of genetic operation is simplified using real encoding, simple mutation operator, and competitive optimization mechanism. jSO algorithm is a new variant of DE algorithm, which is a state-of-the-art algorithm for single objective real-parameter optimization [45], and we, for the first time, introduce it to learn DNM.

The learning algorithm using BP is called DNM + BP, and DNM + BBO, DNM + PSO, DNM + GA, DNM + PBIL, DNM + CSO, DNM + DE, and DNM + jSO are similarly named. The effectiveness, accuracy, and convergence of each algorithm in classification problems are explored and demonstrated using four datasets. The experimental results show that DNM + BBO has fast convergence and high accuracy. At the same time, the classification accuracy of decision tree, KNN, support vector machine (SVM), MLP, and DNM + BBO is compared, and the results show that DNM + BBO has the highest classification accuracy. This paper effectively combines metaheuristic algorithm with dendritic neuron model to establish DNM + BBO, which provides an effective method to solve the classification problem.

2. Model and Learning Algorithms

2.1. Dendritic Neuron Model. The DNM is composed of four layers based on dendrite structure. In the synaptic layer, inputs $x_1, x_2, \ldots, x_n$ of each dendrite are firstly transformed using a sigmoid function. Secondly, in the dendrite layer, the outputs of the first layer are transmitted to a function of multiplication. Thirdly, membrane layer processes the received inputs from the dendrite layer. Finally, the signal from the membrane layer is transformed using another sigmoid function to accomplish the whole process [46]. Figure 1 shows the complete structure of DNM, and below are the details of this model.

2.1.1. Synaptic Layer. A synapse is the connection between neurons. Statistics flow from a synaptic neuron to another, which exhibits a feedforward pattern. The synapse has four connection states, namely, the excitatory connection, the inhibitory connection, constant 0 connection, and constant 1 connection. It depends on changes in the potential of the accepting neuron arising from ionotropic phenomena. The connecting function from the ith ($i = 1, 2, \ldots, n$) synaptic input to the jth ($j = 1, 2, \ldots, m$) synaptic layer is described as follows:

$$Y_{ij} = \frac{1}{1 + e^{-k\left(w_{ij}x_i - \theta_{ij}\right)}}. \tag{1}$$

Equation (1) expresses the transfer function Y_{ij} of the ith input of a synapse x_i varying from 0 to 1. k is constant. w_{ij} and θ_{ij} respectively denote the weight and threshold in synapse.

As for the values of w_{ij} and θ_{ij}, there are four different connections discussed in Figure 2. The X-axis indicates the

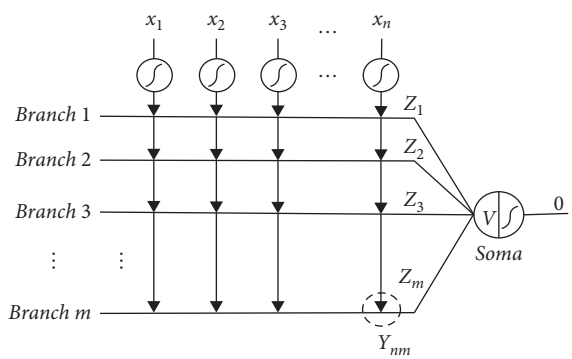


Figure 1: Structure of the dendritic neuron model.

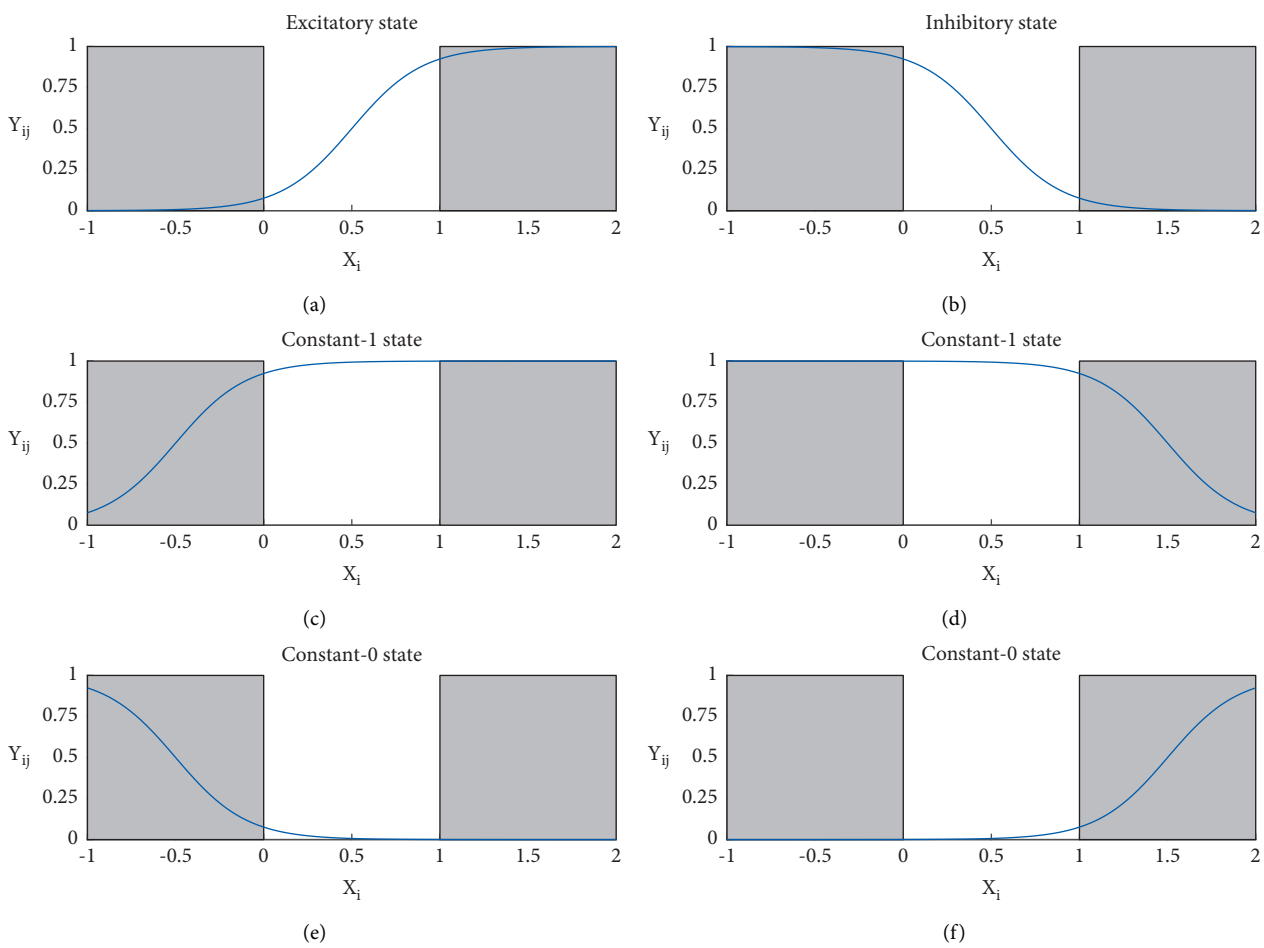


Figure 2: Four different connections in the synaptic layer: (a) excitatory connection, (b) inhibitory connection, (c) constant 1 connection, (d) constant 1 connection, (e) constant 0 connection, and (f) constant 0 connection.

input of the DNM, and the Y-axis indicates the output of the synaptic layer. Since the value of input x is from 0 to 1, only the blank part of each illustration is required to be focused on. The four connections include the following: Figure 2(a) presents excitatory connection, and when $0 < \theta_{ij} < w_{ij}$, the output is proportional to the input. On the contrary, Figure 2(b) depicts the inhibitory connection. As for $w_{ij} < \theta_{ij} < 0$, the output is inversely proportional to the input. Figures 2(c) and 2(d) present constant 1 connection, and when $\theta_{ij} < 0 < w_{ij}$ or $\theta_{ij} < w_{ij} < 0$, regardless of the value of the input, x varies between 0 and 1, and the output is always 1; Figures 2(e) and 2(f) present constant 0 connection, and when $w_{ij} < 0 < \theta_{ij}$ or $0 < w_{ij} < \theta_{ij}$, regardless of the value of the input, x varies between 0 and 1, and the output is always 0.

2.1.2. Dendrite Layer. In this layer, outputs from the synapses are multiplicated altogether. As a method of describing nonlinearity features, multiplication is the first selection due

to its simplicity. In addition, if we take constant 0 or 1 connection as an example, this function is equivalent to the logical AND operator for their similar output values. The output formula for the jth dendrite is as follows:

$$Z_j = \prod_{i=1}^{n} Y_{ij}. \tag{2}$$

2.1.3. Membrane Layer. This layer represents the summary of signals coming from each dendritic branch. The input of the next layer is obtained by a sum function, which resembles a logical OR operator. Then, the processed signal will be transmitted to the soma body. Thus, the output of the membrane layer is as follows:

$$V = \sum_{j=1}^{m} Z_j. \tag{3}$$

2.1.4. Soma Layer. Finally, the received signal in the soma layer is taken as the input of another sigmoid function. The detailed formula for this layer is as follows:

$$O = \frac{1}{1 + e^{-k_s(V-\theta_s)}}, \tag{4}$$

where k_s is a positive constant, and the range of threshold θ_s is [0, 1].

2.2. Backpropagation. Backpropagation (BP) is the gradient descent method [47, 48]. This algorithm contributes to reducing the error between the target output and its real value through neural network training. The error can be expressed as follows:

$$E = \frac{1}{2}(T - O)^2, \tag{5}$$

where T is the target output vector, and O is the actual output vector. By modifying the parameters w_{ij} and θ_{ij} of the DNM model in the process of learning, the error can be decreased. The updated expressions are set as follows:

$$\begin{aligned} \Delta w_{ij}(t) &= \sum_{p=1}^{P} \frac{\partial E_p}{\partial w_{ij}}, \\ \Delta \theta_{ij}(t) &= \sum_{p=1}^{P} \frac{\partial E_p}{\partial \theta_{ij}}, \end{aligned} \tag{6}$$

where E_p is the mean square error. After computing these two increments, we can get values of w_{ij} and θ_{ij} at the next moment through the following:

$$\begin{aligned} w_{ij}(t+1) &= w_{ij}(t) - \eta \Delta w_{ij}(t) \\ \theta_{ij}(t+1) &= \theta_{ij}(t) - \eta \Delta \theta_{ij}(t), \end{aligned} \tag{7}$$

where η denotes the learning rate defined by users. t is a characterization of learning times. Furthermore, the partial differentials of E_p regarding w_{ij} and θ_{ij} are calculated as follows:

$$\begin{aligned} \frac{\partial E_p}{\partial w_{ij}} &= \frac{\partial E_p}{\partial O_p} \frac{\partial O_p}{\partial V} \frac{\partial V}{\partial Z_j} \frac{\partial Z_j}{\partial Y_{ij}} \frac{\partial Y_{ij}}{\partial w_{ij}}, \\ \frac{\partial E_p}{\partial \theta_{ij}} &= \frac{\partial E_p}{\partial O_p} \frac{\partial O_p}{\partial V} \frac{\partial V}{\partial Z_j} \frac{\partial Z_j}{\partial Y_{ij}} \frac{\partial Y_{ij}}{\partial \theta_{ij}}. \end{aligned} \tag{8}$$

The detailed results of the above partial differentials in DNM are given as follows:

$$\begin{aligned} \frac{\partial E_p}{\partial O_p} &= T_p - O_p \\ \frac{\partial O_p}{\partial V} &= \frac{k_s x_i e^{-k_s(w_{ij}x_i - \theta_s)}}{\left(1 + e^{-k_s(w_{ij}x_i - \theta_s)}\right)^2} \\ \frac{\partial V}{\partial Z_j} &= 1 \\ \frac{\partial Z_j}{\partial Y_{ij}} &= \prod_{l=1 an\ dl\ i}^{n} Y_{lj} \\ \frac{\partial Y_{ij}}{\partial w_{ij}} &= \frac{k x_i e^{-k(w_{ij}x_i - \theta_{ij})}}{\left(1 + e^{-k(w_{ij}x_i - \theta_{ij})}\right)^2} \\ \frac{\partial Y_{ij}}{\partial \theta_{ij}} &= \frac{k e^{-k(w_{ij}x_i - \theta_{ij})}}{\left(1 + e^{-k(w_{ij}x_i - \theta_{ij})}\right)^2}. \end{aligned} \tag{9}$$

When computing $\Delta w_{ij}(t)$ and $\Delta\theta_{ij}(t)$, the chain rule is applied, and layer-by-layer calculation is performed throughout the DNM.

2.3. Biogeography-Based Optimization. Biogeography-based optimization derives from biogeography, which investigates the speciation, extinction, and geographical distribution in nature. Each habitat is regarded as a solution and the principal task is to obtain the best one. For convenience, the model introduces mathematics, using high habitat suitability index (HSI) as the degree of fitness among species. And the suitability index variables (SIV) are utilized for describing various aspects of HSI [49]. Procedures using BBO are implemented as follows:

(1) Initializing the integer sequence SIV based on the current habitat H_i ($i = 1, 2, \ldots, n$).

(2) Calculating the HSI of each habitat according to the following formula.

$$HSI(H_i) = \frac{1}{2P} \sum_{p=1}^{P} \left(T_p - O_p\right)^2, \tag{10}$$

where P represents the total number of training samples. T_p is the target vector of the pth sample, and O_p is the actual output vector determined by H_i.

(3) Implementing random selection on the SIV and migration among habitats occurs in the case that the emigration rate and immigration rate are μ_i and λ_i respectively.

$$\mu_i = \frac{E_i}{m}$$
$$\lambda_i = I\left(1 - \frac{i}{m}\right), \tag{11}$$

where E is the emigration rate, I is the maximum immigration rate, and m is the rank of habitat. These two parameters are set as $E = I = 1$ in this research, and λ and μ are constrained as follows:

$$\lambda_i + \mu_i = E. \tag{12}$$

(4) For each habitat H_i, the immigrated his, and the probability Ps_i, it contains the Sth species of habitat that are updated:

$$Ps_i(t + \Delta t) = Ps_i(t)\left(1 - \lambda_i \Delta t - \mu_i \Delta t\right) + Ps_{i-1}\mu_{i-1}\Delta t + Ps_{i+1}\mu_{i+1}\Delta t. \tag{13}$$

If t is small enough to be considered as 0, the following equation can be approximated:

$$Ps_i = \begin{cases} -(\lambda_i + \mu_i)Ps_i + \mu_{i+1}Ps_{i+1} & i = 0 \\ -(\lambda_i + \mu_i)Ps_i + \lambda_{i-1}Ps_{i-1} + \mu_{i+1}Ps_{i+1} & 1 \le i \le n-1 \\ -(\lambda_i + \mu_i)Ps_i + \lambda_{i-1}Ps_{i-1} & i = n \end{cases}. \tag{14}$$

(5) Mutating nonelite habitats according to the mutation rate Pm_i:

$$Pm_i = Pm_{\max}\frac{1 - Ps_i}{Ps_{\max}}, \tag{15}$$

where $Ps_{\max}$ is the maximum value of Ps_i, and $Pm_{\max}$ is the parameter.

(6) Go to Step 2 again and perform the next iteration if needed. Not until the termination criterion is met does this procedure end.

2.4. Particle Swarm Optimization. Particle swarm optimization mimics the search behavior of a flock of birds and consists of particles, each of which represents a possible solution [50]. The solution includes two attributes: speed and position. The former indicates moving rate of each particle, and the latter indicates its moving direction. Each particle moves to the optimal value separately and spontaneously and memorizes the current value for the particle itself (*pbest*). Then, it shares the individual optimal solution with other particles and obtains the global extreme value (*gbest*). All particles update their two attributes according to these two extreme values. PSO is widely adopted as its simple operating process, for instance, MLP [51]. The whole process of PSO being used to search for the optimal values of weights and thresholds in the synaptic layer of DNM can be described as follows:

$$\begin{aligned} X_i &= \left\{x_i^1, x_i^2, \ldots, x_i^m\right\} \\ &= \left\{\omega_{11}, \omega_{12}, \ldots \omega_{MN}, \theta_{11}, \theta_{12}, \ldots \theta_{MN}\right\}, \end{aligned} \tag{16}$$

where X_i ($i = 1, 2, \ldots, Q$) indicates the ith individual in the swarm, and Q denotes the number of particles. Besides, we use the mean square error (MSE) (X_i) to calculate the error of the last layer with output X_i as follows:

$$MSE(X_i) = \frac{1}{2P}\sum_{p=1}^{P}\left(T_p - O_p\right)^2. \tag{17}$$

2.5. Genetic Algorithm. Genetic Algorithm is inspired by natural selection. According to the previous study [30], a set of probable solutions is meant as individuals in optimization study. Good individuals tend to reproduce at a relatively high rate, while poor individuals have a relatively low reproductive rate. With the evolution of population, individuals develop in a healthier direction. However, as GA is a random searching algorithm, chances are that worse individuals are generated from fitter ones in the existing framework. Thus, we adopt the elite strategy to maintain optimized individuals in this scheme. Training DNM can also use GA. Similar to PSO, a chromosome in GA for training DNM can be exhibited using equation (16), where X_i ($i = 1, 2, \ldots, Q$) indicates the ith chromosome in the population, and Q represents the population size. We use single point crossover to update individuals. Moreover, the fitness function is the same as equation (17).

2.6. Population-Based Incremental Learning. Estimation of distribution algorithm (EDA) is a population-based strategy that tracks the statistical information of the candidate solution population for optimization [30, 41, 52]. It uses a solution of discarding at least a part of the population in each generation and using a sample according to the statistic quantity of high-fitness individuals in the current population to generate new populations, and the process is repeated from generation to generation.

Similar to EDA, PBIL is an extension of a univariate marginal distribution algorithm. When it functions as the optimizer of an m-dimensional binary problem, PBIL uses a probability vector p with m dimensions, whose kth value of p describes how likely this element is equal to one. In each generation, a random population is generated using the probability vector p probabilistically. Then, the fitness of each candidate solution is calculated. By adjusting the probability vector, the next generation is more likely to resemble the most suitable individual. After getting this new probability vector, using p to create another candidate solution to the random population, continue this process until the termination requirement is met.

Table 1: Details of the classification datasets.

Classification datasets	# of attributes	# of training samples	# of test samples	# of classes
Banknote authentication	5	960	412	2
Breast cancer	10	489	210	2
Car evaluation	7	1210	518	2
Diabetic retinopathy	17	364	156	2

2.7. Competitive Swarm Optimization. Competitive swarm optimization is a population algorithm used to solve large-scale classification problems, and the velocity of individual movement is not eliminated, same as PSO. It introduces a competitive strategy to make a comparison between two selected particles according to their evaluated results. As only the lost particles can learn to participate in iterating, except for reducing the number of updated particles to $2/N$ [53], it is not necessary to save the excellent solution in search, which can be applied to the effective solution of large-scale classification problems. Below are the operation steps:

(1) N represents the solution at the beginning, and the particle position x_i ($i = 1, 2, \ldots, N$) and velocity v_i ($i = 1, 2, \ldots, N$) of generated particles are initialized.

(2) All solutions are evaluated.

(3) The kth ($k = 1, 2, \ldots, 2/N$) competition for generation t occurs as follows:

(a) Nonrepeating particles N_{k1} and N_{k2} are selected from the undecided particles randomly.

(b) Positions of selected particles of N_{k1} and N_{k2} are compared and evaluated to determine the won particle and the failed particle.

(c) The $x_{l,k}$ of failed particle is updated by the application of its velocity $v_{i,k}$.

$$\begin{aligned} v_{l,k}(t+1) &= R_1(k,t)v_{l,k}(t) \\ &\quad + R_2(k,t)\left[x_{w,k}(t) - x_{l,k}(t)\right] \\ &\quad + \varphi R_3(k,t)\left[\overline{x_k}(t) - x_{l,k}(t)\right] \\ x_{l,k}(t+1) &= x_{l,k}(t) + v_{l,k}(t+1), \end{aligned} \tag{18}$$

where $R_1(k, t)$, $R_2(k, t)$, and $R_3(k, t)$ are vectors with their elements varying from 0 to 1. $\overline{x_k(t)}$ represents the mean position of all particles, and φ represents at which degree the influence of the mean quantity takes effect, which has been advised as the following equation in reference to the existing researches:

$$\begin{cases} \varphi = 0 \; N \le 100 \\ \varphi \in [0.14\log(N) - 0.3,\; 0.27\log(N) - 0.51] \text{ otherwise} \end{cases}. \tag{19}$$

(d) Repeat the three steps above until all particles are identified.

(4) The next iteration starting with step 2 is operated until the parameter t reaches the maximum.

Table 2: Experimental environment of the DNM.

Item	Computing environment
CPU	3.00 GHz intel (R) core (TM) i5-8500
OS	Windows 10 education
RAM	16.0 GB
Software	MATLAB R2018b

2.8. Differential Evolution. Differential evolution is a random search method inspired by biological evolution, and highly fit individuals are preserved through iterations. As a variant of genetic algorithm, it is a global search strategy based on population and adopts real coding, basic mutation from one-to-one difference to simplify the genetic operation. Furthermore, its memory ability enables DE to track the real-time situation and modify the search strategy dynamically. Owing to the prominent global convergence ability and stability, DE is well applied to the solution of complicated optimization problems, which are difficult to resolve using traditional mathematical programming methods. At present, DE has been used in artificial neural network, signal processing, biological information, and other fields.

2.9. jSO. jSO is the latest improved algorithm of differential evolution, which is based on iL-SHADE algorithm [54]. It keeps parameter strategy based on historical memory and linear population size reduction strategy of iL-SHADE. jSO adopts adaptive strategy to improve the mutation coefficient M_F and crossover probability M_{CR}, which are the key parameters of differential evolution algorithm, and the effect is obvious. The mutation strategy of jSO is as follows:

$$\vec{v}_{i,g} = \vec{x}_{i,g} + F_w\left(\vec{x}_{p\text{Best},g} - \vec{x}_{i,g}\right) + F\left(\vec{x}_{r_1,g} - \vec{x}_{r_2,g}\right), \tag{20}$$

$$F_w = \begin{cases} 0.7 * F, nfes < 0.2\text{max\_}nfes \\ 0.8 * F, nfes < 0.4\text{max\_}nfes \\ 1.2 * F, \text{otherwise} \end{cases}, \tag{21}$$

where *nfes* is the current population iteration, and *max_nfes* is the max population iteration.

3. Experiment

In this experiment, four classification problems are used to verify the performance of DNM with the eight learning algorithms mentioned above. Table 1 shows their attributes, number of training samples, number of test samples, and

Table 3: Initial parameters used in learning algorithms.

Algorithm	Parameter	Value
BP	Learning rate	0.01
	Maximum number of generations	1000
BBO	Habitat modification probability	1
	Immigration probability bounds per gene	[0, 1]
	Step size for numerical integration of probabilities	1
	Max immigration and max emigration	1
	Mutation probability	0.005
	Population size	50
	Maximum number of generations	1000
PSO	Acceleration constants	[2]
	Inertia weights	[0.9, 0.5]
	Population size	50
	Maximum number of generations	1000
GA	Selection mechanism	Roulette wheel
	Crossover probability	0.9
	Mutation probability	0.1
	Population size	50
	Maximum number of generations	1000
PBIL	Learning rate	0.05
	Good population member	1
	Bad population member	0
	Elitism parameter	1
	Mutational probability	0.1
	Population size	50
	Maximum number of generations	1000
CSO	The swarm size m	$m\begin{cases}1500\,(d \geq 5000)\\1000\,(d \geq 2000)\\500\,(d \geq 1000)\\250\,(d \geq 500)\\100\text{otherwise}\end{cases}$
	The social factor φ	$\varphi = \begin{cases}0.2\,(d \geq 2000)\\0.15\,(d \geq 1000)\\0.1\,(d \geq 500)\\0\text{otherwise}\end{cases}$
	Population size	50
	Maximum number of generations	1000
DE	Crossover probability	0.9
	Differential weight	0.5
	Population size	50
	Maximum number of generations	1000
jSO	Mutation coefficient M_F	0.3
	Crossover probability M_{CR}	0.5
	Historical memory size H	5
	Population size	50
	Maximum number of generations	1000

Table 4: Reasonable combination of four DNM parameters for four tested problems.

Datasets	M	k	k_s	θ_s
Banknote authentication	20	15	15	0.5
Breast cancer	20	10	5	0.3
Car evaluation	20	15	15	0.5
Diabetic retinopathy	15	5	1	0.1

Table 5: Average accuracy and standard deviation of each learning algorithm.

Datasets	Learning algorithm	Learning's average accuracy (%) ± standard deviation	Test's average accuracy (%) ± standard deviation
Banknote authentication	DNM + BP	63.06 ± 21.63	62.62 ± 21.43
	DNM + BBO	**100 ± 0.00**	**99.43 ± 0.39**
	DNM + PSO	99.84 ± 0.21	98.45 ± 0.79
	DNM + GA	94.11 ± 1.95	92.95 ± 2.42
	DNM + PBIL	93.11 ± 1.06	92.43 ± 2.01
	DNM + CSO	99.94 ± 0.08	99.03 ± 0.42
	DNM + DE	92.05 ± 1.27	91.25 ± 1.86
	DNM + jSO	99.94 ± 0.29	99.39 ± 0.36
Breast cancer	DNM + BP	90.67 ± 3.00	89.84 ± 3.12
	DNM + BBO	**98.92 ± 0.35**	95.97 ± 1.42
	DNM + PSO	97.61 ± 0.58	95.25 ± 1.81
	DNM + GA	97.25 ± 0.64	95.71 ± 1.38
	DNM + PBIL	96.19 ± 0.64	95.10 ± 1.42
	DNM + CSO	98.48 ± 0.34	**96.14 ± 1.23**
	DNM + DE	95.75 ± 0.61	95.25 ± 1.67
	DNM + jSO	97.73 ± 0.68	95.83 ± 1.21
Car evaluation	DNM + BP	63.06 ± 21.63	62.62 ± 21.43
	DNM + BBO	**100 ± 0.00**	**99.43 ± 0.39**
	DNM + PSO	99.84 ± 0.20	98.45 ± 0.79
	DNM + GA	94.11 ± 1.95	92.95 ± 2.42
	DNM + PBIL	93.11 ± 1.06	92.43 ± 2.01
	DNM + CSO	99.94 ± 0.08	99.03 ± 0.42
	DNM + DE	92.05 ± 1.27	91.25 ± 1.86
	DNM + jSO	96.93 ± 1.07	96.49 ± 1.21
Diabetic retinopathy	DNM + BP	38.66 ± 1.05	37.99 ± 2.44
	DNM + BBO	**97.49 ± 0.81**	**94.85 ± 1.75**
	DNM + PSO	90.63 ± 2.30	88.72 ± 3.84
	DNM + GA	87.72 ± 2.84	85.64 ± 3.08
	DNM + PBIL	85.42 ± 2.33	84.21 ± 3.22
	DNM + CSO	94.29 ± 1.41	92.22 ± 2.21
	DNM + DE	82.96 ± 2.48	83.10 ± 2.79
	DNM + jSO	87.49 ± 3.89	86.28 ± 4.90

Table 6: Accuracy of DNM + BBO and other machine learning methods for the datasets of banknote authentication, breast cancer, car evaluation, and diabetic retinopathy.

Datasets	DNM + BBO (%)	Decision tree (%)	SVM (%)	KNN (%)	MLP (%)
Banknote authentication	**99.4**	91.6	97.9	97.2	87.1
Breast cancer	96.0	93.4	**96.6**	94.7	93.8
Car evaluation	**99.4**	92.5	89.8	93.1	82.6
Diabetic retinopathy	**94.9**	89.4	88.3	81.7	87.8

Table 7: Average running time (sec) of each algorithm in different datasets.

Learning algorithm	Banknote authentication	Breast cancer	Car evaluation	Diabetic retinopathy
BP	10	20	3	15
BBO	30	70	8	60
PSO	54	128	9	92
GA	58	133	12	93
PBIL	60	142	12	104
CSO	28	63	6	45
DE	112	251	22	181
jSO	109	113	181	115

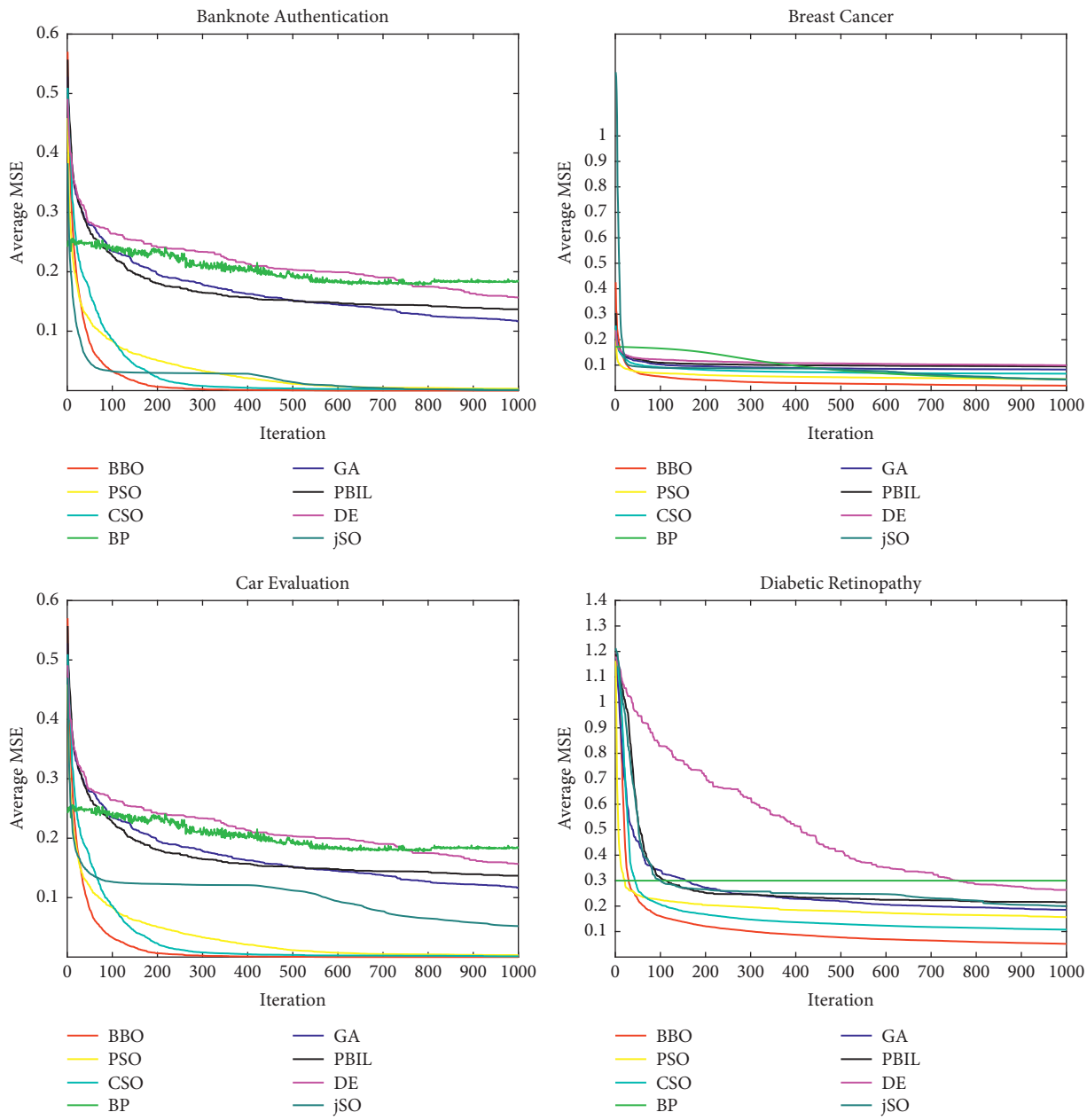


Figure 3: Convergence graphs of the learning algorithms for the datasets of banknote authentication, breast cancer, car evaluation, and diabetic retinopathy, respectively.

number of classes. The classification datasets are acquired from the open datasets of the UCI Machine Learning Repository in various aspects [55].

For each learning algorithm, the maximum generation number is set as 1000. Each data set includes two parts, with learning data accounting for 70% and testing data accounting for 30%. In addition, the characteristics of any classification problem are expressed by numbers with no data error that contain negative numbers and decimal numbers. As the input x of DNM varies from 0 to 1, each characteristic data set is normalized in the corresponding range for the experiment.

All the experimental results are averaged from 30 independent experiments, and the accuracy of the expected output is calculated according to the classification results. Equation (22) is used to calculate the accuracy with true positivity (TP), false positive (FP), true negative (TN), and false negative (FN). Besides, MSE is utilized as the evaluation function using equation (17) for each learning algorithm.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \times 100\%. \tag{22}$$

As for the experimental equipment and operating rate, the experimental environment is shown in Table 2. The design of experiment adopts a statistical strategy for the effective analysis of large combinations using orthogonal arrays based on Latin square. The mentioned eight learning algorithms are tested under above situations, and owing to

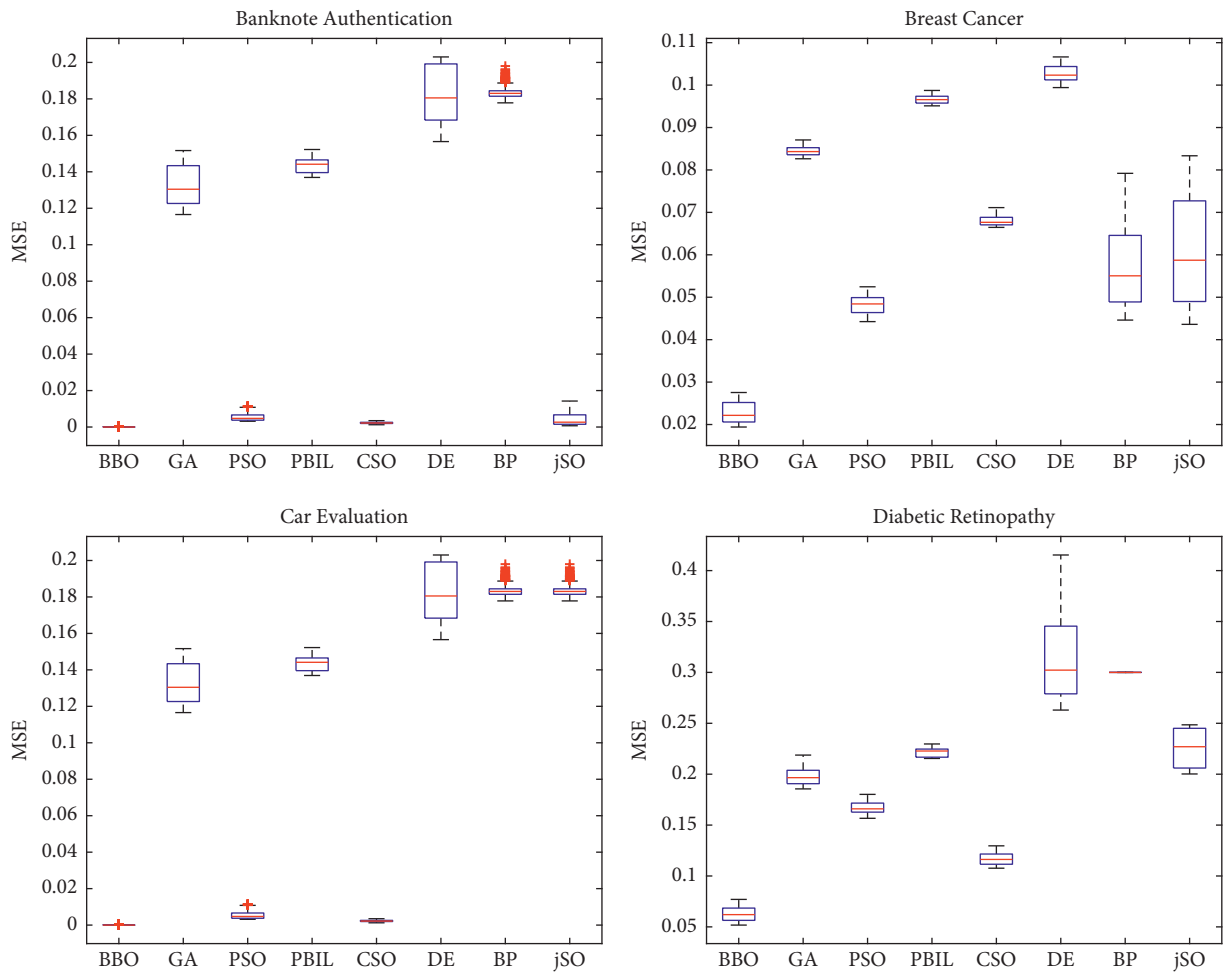


Figure 4: Solution distribution of DNM for the datasets of banknote authentication, breast cancer, car evaluation, and diabetic retinopathy, respectively.

the adoption of orthogonal arrays, the number of experiments can be greatly reduced by the relationship between the factors and levels.

From the previous research, the performance of the learning algorithm can be significantly enhanced by carefully selecting parameter values in some preliminary experiments [56–58]. According to experience and algorithm characteristics, Table 3 shows parameters setting for each algorithm. The population size is set to 50, and the maximum number of generations of each algorithm is 1000 uniformly, while other parameters are set empirically according to characteristics of each algorithm.

For obtaining the optimal performance of DNM, user-defined parameters are well worth investigating. There are four key parameters in DNM, that is, the number of dendrites in the model (M), the synaptic parameter in the connecting sigmoid function (k), and two soma parameters (k_s and θ_s) in the output sigmoid function.

The reasonable combination of four DNM parameters is obtained by Taguchi method [59–61]. It scans a portion of possible combinations among factors rather than the whole combination, resulting in minimal experimental runs and optimal estimation of factors during execution [62, 63]. Referring to relevant previous study and research experience, the hierarchy number of four factors is set as follows: five levels for $M \in \{3, 5, 10, 15, 20\}$; five levels for $k \in \{1, 5, 10, 15, 20\}$; five levels for $k_s \in \{1, 5, 10, 15, 25\}$; and five levels for $\theta_s \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. Different from full factor analysis, which requires $5^4 = 625$ trials, the orthogonal array can obviously reduce the number of experiments and time cost. Hence, the orthogonal array $L_{25}(5^4)$, which involves only 25 experiments, is utilized in this time.

The supplementary material (available here) summarizes the experimental results, where MSE represents the mean square error values of the eight learning algorithms (i.e., BP, BBO, PSO, GA, PBIL, CSO, DE, and jSO) for four datasets. According to the experimental results of each dataset, we obtained acceptable user-defined parameter settings respectively, as shown in Table 4.

4. Result

The average accuracy and standard deviation of learning algorithms in each dataset are summarized in Table 5. It is obvious that regardless of which dataset, DNM + BBO achieves the highest accuracy among all the comparison

Table 8: Average accuracy, standard deviation, average MSE, and corresponding optimal parameters of each learning algorithm in different datasets.

Learning algorithm	Datasets	M	k	k_s	θ_s	Average accuracy (%) ± standard deviation	Average MSE
BP	Banknote authentication	5	5	1	0.1	96.93 ± 2.05	3.33E − 02
	Breast cancer	20	10	5	0.3	89.84 ± 3.13	4.46E − 02
	Car evaluation	15	5	1	0.1	98.31 ± 1.03	2.80E − 02
	Diabetic retinopathy	20	10	5	0.3	66.32 ± 23.63	1.13E − 01
BBO	Banknote authentication	20	15	15	0.5	99.43 ± 0.39	3.44E − 07
	Breast cancer	20	10	5	0.3	95.97 ± 1.42	1.94E − 02
	Car evaluation	20	15	15	0.5	99.43 ± 0.39	3.44E − 07
	Diabetic retinopathy	15	5	1	0.1	94.85 ± 1.75	5.18E − 02
PSO	Banknote authentication	20	1	25	0.9	98.79 ± 0.75	2.76E − 03
	Breast cancer	20	1	25	0.9	95.24 ± 1.64	4.13E − 02
	Car evaluation	20	1	25	0.9	98.79 ± 0.75	2.76E − 03
	Diabetic retinopathy	15	5	1	0.1	88.72 ± 3.84	1.57E − 01
GA	Banknote authentication	10	20	15	0.7	94.89 ± 2.03	7.82E − 02
	Breast cancer	20	15	15	0.5	95.22 ± 1.52	5.54E − 02
	Car evaluation	10	20	15	0.7	94.90 ± 2.03	7.82E − 02
	Diabetic retinopathy	15	5	1	0.1	85.64 ± 3.08	1.86E − 01
PBIL	Banknote authentication	5	1	25	0.9	94.09 ± 1.74	9.72E − 02
	Breast cancer	20	1	25	0.9	94.30 ± 1.27	7.41E − 02
	Car evaluation	5	15	15	0.5	94.09 ± 1.74	9.72E − 02
	Diabetic retinopathy	15	5	1	0.1	84.21 ± 3.22	2.07E − 01
CSO	Banknote authentication	20	1	25	0.9	98.91 ± 0.66	8.98E − 04
	Breast cancer	20	1	25	0.9	95.49 ± 1.19	3.14E − 02
	Car evaluation	20	1	25	0.9	98.91 ± 0.66	8.98E − 04
	Diabetic retinopathy	15	5	1	0.1	92.22 ± 2.21	1.14E − 01
DE	Banknote authentication	3	15	10	0.5	93.50 ± 1.70	1.10E − 01
	Breast cancer	3	10	5	0.3	94.75 ± 0.81	6.86E − 02
	Car evaluation	3	15	10	0.5	93.50 ± 1.69	1.10E − 01
	Diabetic retinopathy	15	10	5	0.3	90.89 ± 3.72	9.31E − 02
jSO	Banknote authentication	15	1	25	0.9	99.39 ± 0.36	6.95E − 04
	Breast cancer	3	1	25	0.9	95.83 ± 1.21	4.36E − 02
	Car evaluation	5	1	25	0.9	96.49 ± 1.21	5.21E − 02
	Diabetic retinopathy	20	15	15	0.5	86.28 ± 4.90	2.00E − 01

object, and some even reach 100%, while DNM + BP is the lowest. Otherwise, the accuracy of DNM + CSO is also higher, but inferior to that of DNM + BBO and higher than that of DNM + PSO. Moreover, the accuracy of DNM + GA, DNM + jSO, DNM + PBIL, and DNM + DE with little difference is relatively common.

Furthermore, the upper limit of m is set as 20 in this experiment, but both optimum parameters m of DNM + BBO and DNM + CSO are 15, three quarters of the upper limit, with a high accuracy of classification as shown in Table 5. As a result, for classification problems, DNM learning by metaheuristics may not require an extremely large number of m for problems with a small number of features.

Table 6 shows the accuracy comparison results of DNM + BBO and other common machine learning classification methods on four datasets, and DNM + BBO has obvious advantages. Using the Machine Learning Toolbox of MATLAB to realize decision tree, SVM, and KNN. Table 7 shows the average running time (sec) of each algorithm in four datasets. BP runs the fastest with the worst classification performance. Among metaheuristic algorithms, CSO is the best, and BBO is the second, while DE and jSO have bad performance.

Figure 3 presents the average convergence graph of each learning algorithm for the datasets of banknote authentication, breast cancer, car evaluation, and diabetic retinopathy, respectively. It is evident that the value of each learning algorithm converges to the final iteration time, but BBO converges the fastest in all cases. The multipoint search BBO maintains an elite habitat, changing the solution in each iteration, which produces a higher number of new candidate optimal solutions than any other learning algorithms.

By deriving a new solution from a candidate optimal solution at a certain time, the convergence rate of the high-quality solution can be accelerated. As a result, BBO has basically converged to the minimum value of MSE even in the 200 iterations in case of banknote authentication and car evaluation.

On the contrary, the MSE of BP is not significantly affected by the local solution in all datasets, indicating that BP is easily trapped in the local minima. Compared with the multipoint search method, BP has the disadvantages of insufficient problem orientation and easy deviation from the local solution. Therefore, different from other algorithms, we believe that BBO has the characteristics of obtaining smaller

Table 9: Overall statistical comparison obtained by Friedman test of seven learning algorithms for four datasets.

Algorithm	Average ranking	p_{Bonf}	$\alpha = 0.05$
BBO	1.53	—	—
CSO	2.46	0.088	Yes
PSO	4.38	0.002	Yes
GA	4.36	0.032	Yes
jSO	4.41	0.047	Yes
BP	5.1	0.057	Yes
PBIL	5.43	0.001	Yes
DE	5.58	0.006	Yes

MSE under fewer iterations, which emphasizes the unique advantages of DNM over the state-of-the-art methods.

Moreover, Figure 4 depicts the average solution distribution of 30 independent runs of each learning algorithm for four datasets, respectively. BBO has the best stability that often finds stable solutions, whereas GA, DE, jSO, or BP usually varies widely across different runs. In addition, it can be easily found that the minimum MSE for each dataset is BBO, and the maximum MSE for each dataset is BP.

According to the supplementary material (available here), the average accuracy, standard deviation, average MSE, and corresponding optimal parameter of each learning algorithm in different datasets are summarized in Table 8.

Table 9 shows the overall statistical results of eight learning algorithms for four problems via the Friedman test at the level of $\alpha = 0.05$ with the application of Bonferroni-Dunn procedures. Friedman test indicates whether there is a difference in the average rank between various ordinal variables. Post hoc test is Bonferroni-Dunn procedure, whose adjusted p value is p_{Bonf}. The result shows that BBO has excellent robustness in each problem and performs better than other methods in terms of average accuracy, standard deviation, and average MSE. This proves that BBO has higher stability and is not easily affected by the problem. Furthermore, CSO is the algorithm with the best results except BBO.

On the contrary, results of BP are the worst, which vary greatly with different datasets, indicating that its accuracy, reliability, and stability are not good and are easily affected by problems. Consequently, the accuracy of any learning algorithm will be biased according to the compatibility of the problem and the combination of parameters, and in terms of convergence and stability of MSE, especially for convergence speed, BBO is reliable and has obvious advantages.

5. Discussion

The experimental results show that DNM + BBO has significant advantages over the other seven optimization algorithms (BP, PSO, GA, PBIL, CSO, DE, and jSO) and other machine learning algorithms (decision tree, SVM, KNN, and MLP). This is due to the mechanism of BBO algorithm, through interspecies migration and intraspecies mutation, the feature information of different habitats has changed, and the dominant features of habitats have been shared; it avoids a large number of local solutions in DNM training. Although DNM + BBO shows great potential in classification problems, there are many challenges in practical applications, such as redundancy in data and more irrelevant information, which reduce the performance and increase the computational complexity of machine learning classification algorithms. For high-dimensional heterogeneous data, we will continue to solve the problems of DNM in future research.

6. Conclusion

With the advent of the era of big data, researches on high-precision models with simple structure and low cost are developing rapidly in solving complicated problems. In this paper, considering the synaptic nonlinearity, a new learning algorithm based on the DNM model is put forward to solve complex classification optimization problems. Adopting factor allocation and orthogonal array, eight learning algorithms (i.e., BP, BBO, PSO, GA, PBIL, CSO, DE, and jSO) are used to train the DNM systematically for four datasets. And the effectiveness, stability, classification accuracy, and convergence speed of these algorithms are compared and demonstrated for such problems.

The experimental results show that BP cannot find the global optimal weight and threshold since its inherent local minimum trap problem, and its effect and accuracy are extremely limited. And the performance of BBO is the most competitive. No matter what kind of dataset is used, the stability, accuracy, and convergence speed of BBO are the most excellent and obviously superior to those of other algorithms. Moreover, the comprehensive performance of CSO is the best except for BBO. The performance of PSO is second only to that of CSO. For GA, PBIL, DE, and state-of-the-art jSO, although they are slightly better than BP, the results vary greatly, while the datasets are different. In addition, compared with common machine learning methods such as decision tree, support vector machine, k-nearest neighbor, and artificial neural networks, dendritic neuron model trained by biogeography-based optimization has significant advantages.

Therefore, this paper combines metaheuristic algorithm and dendrite neuron model effectively, and DNM + BBO is established in this study, especially as a powerful algorithm to solve classification and optimization problems. In the future, we will further explore and try to expand the output range of DNM, apply it to a wider range of fields to solve other different problems, and continue to pursue improving its performance [64].

Data Availability

The datasets of this paper are obtained from the following website: https://archive.ics.uci.edu/ml/index.php.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (NSFC) under Grant nos. 12105120, 21805106, 62102169 and 72174079, the Natural Science Foundation of Jiangsu Province under Grant BK20181073, the MOE Key Laboratory of TianQin Project, Sun Yat-sen University, and the Open Fund Project of Jiangsu Institute of Marine Resources Development under Grant nos. JSIMR202018.

Supplementary Materials

Experimental results of eight learning algorithms (i.e., BP, BBO, PSO, GA, PBIL, CSO, DE, and jSO) over 30 independent runs for four datasets. Tables S1–S4 show the average accuracy of each learning algorithm for the full parameter combination in each dataset. Furthermore, Tables S5–S8 depict the parameter sensitivity results based on factor assignment and orthogonal array of each learning algorithm for each dataset. (*Supplementary Materials*)

References

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[2] S. Haykin, *Neural Networks*, A Comprehensive Foundation, Upper Saddle River, NJ, USA, 1994.

[3] Y. Todo, H. Tamura, K. Yamashita, and Z. Tang, "Unsupervised learnable neuron model with nonlinear interaction on dendrites," *Neural Networks*, vol. 60, pp. 96–103, 2014.

[4] Y. G. Lei, B. Yang, X. Jiang, F. Jia, N. Li, and A. K. Nandi, "Applications of machine learning to machine fault diagnosis: a review and roadmap," *Mechanical Systems and Signal Processing*, vol. 138, Article ID 106587, 2020.

[5] C. Koch, T. Poggio, and V. Torre, "Nonlinear interactions in a dendritic tree: localization, timing, and role in information processing," *Proceedings of the National Academy of Sciences*, vol. 80, no. 9, pp. 2799–2802, 1983.

[6] R. C. Paolicelli, G. Bolasco, F. Pagani et al., "Synaptic pruning by microglia is necessary for normal brain development," *Science*, vol. 333, no. 6048, pp. 1456–1458, 2011.

[7] L. K. Low and H. J. Cheng, "Axon pruning: an essential step underlying the developmental plasticity of neuronal connections," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 361, no. 1473, pp. 1531–1544, 2006.

[8] Z. Zhang, Z. Li, Y. Zhang, Y. Luo, and Y. Li, "Neural-dynamic-method based dual-arm CMG scheme with time-varying constraints applied to humanoid robots," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 12, pp. 3251–3262, 2015.

[9] H. T. He, S. C. Gao, T. Jin, S. Sato, and X. Y. Zhang, "A seasonal-trend decomposition-based dendritic neuron model for financial time series prediction," *Applied Soft Computing*, vol. 108, Article ID 107488, 2021.

[10] W. Chen, J. Sun, S. Gao, J. J. Cheng, J. Wang, and Y. Todo, "Using a single dendritic neuron to forecast tourist arrivals to Japan," *IEICE - Transactions on Info and Systems*, vol. E100.D, no. 1, pp. 190–202, 2017.

[11] S. C. Gao, M. C. Zhou, Y. R. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 2, pp. 601–614, 2019.

[12] S. C. Gao, M. Z. Zhou, Z. Q. Wang et al., "Fully Complex-Valued Dendritic Neuron Model," *IEEE transactions on neural networks and learning systems*, pp. 1–14, 2021, In Press.

[13] Y. Yu, Z. Y. Lei, Y. R. Wang, T. Zhang, C. Peng, and S. Gao, "Improving dendritic neuron model with dynamic scale-free network-based differential evolution," *IEEE/CAA Journal of automatica sinica*, vol. 9, no. 1, pp. 99–110, 2022.

[14] Z. Xu, Z. Q. Wang, J. Y. Li, T. Jin, X. Meng, and S. Gao, "Dendritic neuron model trained by information feedback-enhanced differential evolution algorithm for classification," *Knowledge-Based Systems*, vol. 233, Article ID 107536, 2021.

[15] R. D. Cazé, S. Jarvis, A. J. Foust, and S. R. Schultz, "Dendrites enable a robust mechanism for neuronal stimulus selectivity," *Neural Computation*, vol. 29, no. 9, pp. 2511–2527, 2017.

[16] S. M. Almufti, R. Boya Marqas, and V. Ashqi Saeed, "Taxonomy of bio-inspired optimization algorithms," *Journal of Advanced Computer Science & Technology*, vol. 8, no. 2, 23 pages, 2019.

[17] Q. Kang, B. Huang, and M. Zhou, "Dynamic behavior of artificial Hodgkin–Huxley neuron model subject to additive noise," *IEEE Transactions on Cybernetics*, vol. 46, no. 9, pp. 2083–2093, 2016.

[18] Z. J. Sha, L. Hu, Y. Todo, J. Ji, S. Gao, and Z. Tang, "A breast cancer classifier using a neuron model with dendritic nonlinearity," *IEICE - Transactions on Info and Systems*, vol. E98.D, no. 7, pp. 1365–1376, 2015.

[19] R. Legenstein and W. Maass, "Branch-specific plasticity enables selforganization of nonlinear computation in single neurons," *Journal of Neuroscience*, vol. 31, no. 30, pp. 10787–10802, 2011.

[20] J. Bono, K. A. Wilmes, and C. Clopath, "Modelling plasticity in dendrites: from single cells to networks," *Current Opinion in Neurobiology*, vol. 46, pp. 136–141, 2017.

[21] D. B. Jia, K. Yanagisawa, Y. Ono et al., "Multiwindow non-harmonic analysis method for gravitational waves," *IEEE Access*, vol. 6, pp. 48645–48655, 2018.

[22] D. B. Jia, K. Yanagisawa, M. Hasegawa et al., "Time-frequency-based non-harmonic analysis to reduce line noise impact for LIGO observation system," *Astronomy and computing*, vol. 25, pp. 238–246, 2018.

[23] W. X. Xu, C. H. Li, Y. X. Dou et al., "Optimizing the weights and thresholds in dendritic neuron model using the whale optimization algorithm," *Journal of Physics: Conference Series*, vol. 2025, no. 1, Article ID 012037, 2021.

[24] D. B. Jia, H. W. Dai, Y. Takashima et al., "EEG Processing in internet of medical things using non-harmonic analysis: application and evolution for SSVEP responses," *IEEE Access*, vol. 7, pp. 11318–11327, 2019.

[25] M. M. Ghiasi, S. Zendehboudi, and A. A. Mohsenipour, "Decision tree-based diagnosis of coronary artery disease: CART model," *Computer Methods and Programs in Biomedicine*, vol. 192, Article ID 105400, 2020.

[26] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Let a biogeography based optimizer train your multi-layer perceptron," *Information sciences*, vol. 269, pp. 188–209, 2014.

[27] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning Internal Representations by Error Propagation*, University of California, San Diego, CA, USA, 1985.

[28] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: perceptron, madaline, and backpropagation," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1415–1442, 1990.

[29] Y. Yu, S. C. Gao, Y. Wang, and Y. Todo, "Global optimum-based search differential evolution," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 2, pp. 379–394, 2019.

[30] T. V. Mathew, *Genetic Algorithm*, IIT Bombay, Adi Shankaracharya Marg, Powai, Mumbai, Maharashtra, India, 2012.

[31] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, no. 2, pp. 65–85, 1994.

[32] B. Jafrasteh and N. Fathianpour, "A hybrid simultaneous perturbation artificial bee colony and back-propagation algorithm for training a local linear radial basis neural network on ore grade estimation," *Neurocomputing*, vol. 235, pp. 217–227, 2017.

[33] S. Dan, "Biogeography-based optimization," *IEEE transaction on evolutionary computation*, vol. 12, pp. 702–713, 2008.

[34] R. M. Li, Y. F. Huang, and J. Wang, "Long-term traffic volume prediction based on K-means Gaussian interval type-2 fuzzy sets," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, pp. 1–8, 2019.

[35] D. B. Jia, Y. Fujishita, C. H. Li, Y. Todo, and H. W. Dai, "Validation of large-scale classification problem in dendritic neuron model using particle antagonism mechanism," *Electronics*, vol. 9, no. 5, 792 pages, 2020.

[36] F. Wang, H. Zhang, and A. Zhou, "A particle swarm optimization algorithm for mixed-variable optimization problems," *Swarm and Evolutionary Computation*, vol. 60, Article ID 100808, 2021.

[37] D. B. Jia, C. H. Li, Q. Liu et al., "Application and evolution for neural network and signal processing in large-scale systems," *Complexity*, vol. 2021, no. 7, Article ID 6618833, 2021.

[38] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Transaction on cybernetics*, vol. 45, no. 2, pp. 191–204, 2014.

[39] A. H. Khan, X. W. Cao, S. Li, V. N. Katsikis, and L. Liao, "BAS-ADAM: an ADAM based approach to improve the performance of beetle antennae search optimizer," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 2, pp. 461–471, 2020.

[40] B. A. Norman and A. E Smith, "Random keys genetic algorithm with adaptive penalty function for optimization of constrained facility layout problems," in *Proceedings of 1997 IEEE International Conference on Evolutionary Computation*, pp. 407–411, Indianapolis, IN, USA, 1997.

[41] Y. Li, X. Feng, and G. Wang, "Application of population based incremental learning algorithm in satellite mission planning," in *International Conference on Wireless and Satellite Systems*Springer, Cham, 2020.

[42] L. Grippo, A. Manno, and M. Sciandrone, "Decomposition techniques for multilayer perceptron training," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 11, pp. 2146–2159, 2016.

[43] A. Kaveh and N. Farhoudi, "A new optimization method: dolphin echolocation," *Advances in Engineering Software*, vol. 59, pp. 53–70, 2013.

[44] R. Soto, B. Crawford, R. Olivares et al., "A reactive population approach on the dolphin echolocation algorithm for solving cell manufacturing systems," *Mathematics*, vol. 8, no. 9, 1389 pages, 2020.

[45] J. Brest, M. S. Maučec, and B. Bošković, "Single objective real-parameter optimization: algorithm jSO," in *Proceedings of the 2017 IEEE congress on Evolutionary Computation (CEC)*, pp. 1311–1318, Donostia, Spain, June 2017.

[46] X. X. Qian, Y. R. Wang, S. C. Gao, and K. Todo, "Mr2DNM: A Novel Mutual Information-Based Dendritic Neuron Model," *Computational intelligence and neuroscience*, vol. 2019, Article ID 7362931, 2019.

[47] R. Chakraborty and N. R. Pal, "Feature selection using a neural framework with controlled redundancy," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 1, pp. 35–50, 2015.

[48] R. V. Gandhi and D. M. Adhyaru, "Takagi-Sugeno fuzzy regulator design for nonlinear and unstable systems using negative absolute eigenvalue approach," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 2, pp. 482–493, 2020.

[49] F. Gabbiani, H. G. Krapp, C. Koch, and G. Laurent, "Multiplicative computation in a visual neuron sensitive to looming," *Nature*, vol. 420, no. 6913, pp. 320–324, 2002.

[50] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory MHS'95," in *Proceedings of the Sixth International symposium on Micro Machine and Human Science*, pp. 39–43, Nagoya, Japan, October 1995.

[51] C. F. Juang, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 2, pp. 997–1006, 2004.

[52] P. Larrañaga, "A Review on Estimation of Distribution Algorithms," *Estimation Of Distribution Algorithms*, vol. 2, pp. 57–100, 2002.

[53] W. X. Xu, C. H. Li, D. B. Jia et al., "A dendritic neuron model for disease prediction," in *Proceedings of the 2021 IEEE International Conference on Advances in Electrical Engineering and Computer Applications*, pp. 1118–1122, Dalian, China, August 2021.

[54] J. Brest, M. S. Maučec, and B. Bošković, "iL-SHADE: Improved L-SHADE Algorithm for Single Objective Real-Parameter Optimization," in *Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1188–1195, Vancouver, BC, Canada, July 2016.

[55] Uci Machine Learning Repository, 2022, https://archive.ics.uci.edu/ml/index.php.

[56] J. J. Cheng, G. Y. Yuan, M. C. Zhou et al., "Accessibility analysis and modeling for IoV in an urban scene," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4246–4256, 2020.

[57] J. J. Cheng, G. Y. Yuan, M. C. Zhou, S. Gao, C. Liu, and H. Duan, "A fluid mechanics-based data flow model to estimate VANET capacity," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, pp. 2603–2614, 2020.

[58] J. J. Cheng, X. Wu, M. C. Zhou, S. C. Gao, Z. H. Huang, and C. Liu, "A novel method for detecting new overlapping community in complex evolving networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 9, pp. 1832–1844, 2019.

[59] G. Taguchi and R. Jugulum, *Computer-based Robust Engineering: Essentials for DFSS*, ASQ Quality Press, Mexico, 2004.

[60] J. J. Cheng, M. J. Chen, M. C. Zhou, S. C. Gao, C. N. Liu, and C Liu, "Overlapping community change-point detection in an evolving network," *IEEE transactions on big data*, vol. 6, no. 1, pp. 189–200, 2020.

[61] J. Sun, S. C. Gao, H. W. Dai, J. Cheng, M. Zhou, and J. Wang, "Bi-objective elite differential evolution algorithm for multivalued logic networks," *IEEE Transactions on Cybernetics*, vol. 50, no. 1, pp. 233–246, 2020.

[62] J. F. C. Khaw, B. S. Lim, and L. E. Lim, "Optimal design of neural networks using the Taguchi method," *Neurocomputing*, vol. 7, no. 3, pp. 225–245, 1995.

[63] S. C. Gao, Y. Yu, Y. Wang, J. Wang, J. Cheng, and M. Zhou, "Chaotic local search-based differential evolution algorithms for optimization," *IEEE transactions on systems, man, and cybernetics: Systems*, vol. 51, no. 6, pp. 3954–3967, 2021.

[64] W. X. Xu, D. B. Jia, Z. M. Zhong, C. H. Li, and Z. X. Xu, "Intelligent dendritic neural model for classification problems," *Symmetry*, vol. 14, no. 1, 11 pages, 2021.

Hindawi
Discrete Dynamics in Nature and Society
Volume 2022, Article ID 2021535, 12 pages
https://doi.org/10.1155/2022/2021535

Research Article

Intelligent Warehouse Robot Scheduling System Using a Modified Nondominated Sorting Algorithm

Jia Ma,[1] **Shujun Yang**,[2] **and Hao Jing**[1]

[1]*College of Economics and Management, Shenyang Aerospace University, Shenyang 110136, China*
[2]*College of Software, Northeastern University, Shenyang 110819, China*

Correspondence should be addressed to Shujun Yang; yangshujun@stumail.neu.edu.cn

Received 29 March 2022; Revised 24 April 2022; Accepted 13 May 2022; Published 15 June 2022

Academic Editor: Shi Cheng

In intelligent warehouse, the problem of transporting goods in intelligent warehouse is becoming increasingly complex, and the traditional way of automatically guiding vehicles (AGVs) is inefficient, so automated robot systems are introduced into intelligent warehouses. In this paper, a task assignment model for robots is presented with the transportation problem of robots in intelligent warehouse as the research background. To solve the robot task assignment problem in intelligent warehouse, a novel Pareto-based multiobjective optimization algorithm (MOEA) is proposed, and the aggregation function is invoked to replace the crowding distance; the brain storm operator is used for crossover and mutation. Finally, the ability of the algorithm to solve the benchmark test problem suite and real-world problems is experimentally confirmed.

1. Introduction

Recently, the logistics industry is facing more fierce competition, forcing the logistics industry to adopt cheap and efficient automatic robots to replace the traditional automatic guidance vehicles (AGVs) and adopt new intelligent robot scheduling system to reduce operating costs and improve storage efficiency [1]. From the perspective of warehouse management, it is important to quickly transport goods in the warehouse to reduce the production and transaction cycles. In other words, the basic task of intelligent warehouse multirobot scheduling system is to store, transport, and extract goods [2]. Therefore, the task allocation program needs to allocate tasks rationally and effectively. In the process of multirobot task scheduling, a group of tasks need to be scheduled in an optimal way (for example, allocation and execution).

At present, there are some metaheuristic algorithms to handle the robot allocation problems. Whale optimization algorithm (WOA) is used to handle mobile robot assignment in the intelligent manufacturing system [3]. Hyperheuristic algorithms based on stigmergy and flocking are used to solve the robot search problem in the unmanned space [4]. A generalized graph-based heuristic algorithm is used to solve a dynamic route planning problem during robot movement [5]. *Sam* uses metaheuristic algorithm to provide carton manufacturers with an optimal robot transportation strategy for cases with more autonomous robots [6]. *Faiza* proposes a (Grey wolf optimization and particle swarm optimization) PSO-GWO algorithm to solve the problem of mobile robot avoidance of obstacles [7]. *Liu* proposes two dynamic order planning algorithms to reduce the order delay problem in smart warehouses [8]. *Tan* solves the vertical picking problem in the warehouse using the PSO algorithm [9].

However, most recent work on task assignment in intelligent warehouse has focused only on single-objective minimization, i.e., it has considered only the minimization of overall time and not the minimization of individual autonomous robot time [10]. For this reason, we build optimization problems based on the characteristics of robot task assignment in intelligent warehouse, which includes two objective functions; the first objective function is minimization of overall time for the robots to perform tasks and the other objective function is to maximize the time a single robot spends performing tasks. Therefore, this is a

multiobjective optimization problem (MOP) [11], where multiple objectives need to be optimized simultaneously. Meanwhile, with the increase of orders in warehouse, the scheduling of multiple robots in the intelligent warehouse will become more difficult [12].

For real-world MOPs with multiple attributes, the multiobjective optimization algorithm (MOEA) appears [13–15]. Among these MOEAs, Pareto-based method has always been an effective method to solve MOPs [16], and the most representative algorithms are NSGA-II [17] and SPEA2 [18]. In these algorithms, nondominated sorting has great advantages in solving 2,3-objective optimization problems, it allows nondominated stratification of the population and then selects individuals which satisfy the conditions to the next generation.

In summary, a novel algorithm which uses nondominated sorting and maximin aggregation function was proposed for the task allocation problem in intelligent warehouse. In this algorithm, nondominated sorting is used to quickly select solutions, and maximin function and one-by-one strategy are used to maintain the uniform distribution of the population. Finally, brain storm operator is used to generate new individuals [19]. Some effective optimization properties of maximin aggregation function will be introduced in the following sections. For the proposed algorithm, the maximin aggregation function can better replace the crowdedness distance to evaluate the contribution of individuals in the same nondominated layer to the new population, the one-by-one comparison strategy can more accurately select a more suitable new solution from the candidate solutions, and the randomness of the brain storm optimization operator helps the algorithm to jump out of the local optimum and enhance the global search ability of the algorithm.

The contributions of this paper are as follows:

(i) The task assignment problem for the intelligent warehouse incorporating minimization of overall multirobot time and maximization of individual robot time is proposed. For this task scheduling model, a different crossover mutation operator is adopted.

(ii) For this multirobot scheduling problem, we propose a new Pareto-based algorithm, which also uses maximin function and one-by-one comparison strategy as a supplement, and uses Brain Storm operator to select the parents for crossover and mutation.

(iii) The results of simulation experiments on ZDT and DTLZ benchmark test suites and task assignment problem in intelligent warehouse verify the excellent performance of the algorithm in solving MOPs.

The other sections are as follows. Section 2 presents a mathematical model for robot task assignment in the intelligent warehouse. In Section 3, related technologies and the proposed algorithm MB-NSGA-II are shown. The experimental results are summarized and analyzed in Section 4. Finally, Section 5 gives the conclusion.

2. Problem Formulation

2.1. Background Description. For the intelligent warehouse, a batch of goods after delivery to the warehouse, the manager will provide a list of tasks to the intelligent warehouse system based on demand [20]. What the task assignment in the intelligent warehouse needs to do is to reasonably allocate this group of task sequences to the autonomous robots. Among them, the tasks are divided into the following categories: (1) Inbound task, transporting goods to the warehouse; (2) Transportation task is to move goods from one shelf to another according to the needs of the manager; (3) Outbound task, transporting goods out of the warehouse.

The two-dimensional plane diagram of the intelligent warehouse is shown in Figure 1, in which the neatly arranged look grid represents the shelves. The white cells in the shelves indicate that the shelves are empty because there are no goods stored on them. The automatic robot walks in the aisle to reach the position of the shelf to transport goods. In this way, the information such as the shelf of the intelligent warehouse and the walking range of the robot can be clearly displayed. In the example Figure 1, all goods enter the warehouse from the bottom right and leave the warehouse from the top left.

For the robot task assignment issues, the two objective functions of the task assignment model are minimizing the total time for robots to perform tasks and minimizing the time for a single robot to perform tasks, respectively. Assuming that there are m intact robots that can move freely and n tasks to be assigned in the intelligent warehouse, the robots can assign inbound tasks, transportation tasks, and outbound tasks, respectively. When a robot completes a series of tasks, it will encounter the following situations. For the inbound task and transportation task, the coordinates of task t_i are assumed to be (x_i, y_i). When the time consumption between task and task is not considered, the time consumption of robot executing task t_i is $TC_{\text{inbound}}(t_i)$. Therefore, time required for outbound task $TC_{\text{outbound}}(t_i)$ is

$$\mathrm{TC}_{\text{outbound}}(t_i) = \left(\left|x_i - x_{\text{in}}\right| + \left|y_i\right|\right). \tag{1}$$

The time consumption of the inbound task $TC_{\text{inbound}}(t_i)$ is

$$\mathrm{TC}_{\text{inbound}}(t_i) = \left(\left|x_i\right| + \left|y_i - y_{\text{out}}\right|\right). \tag{2}$$

Unlike the inbound task and the outbound task, the transportation task is different. In the transportation task, assuming that task t_i needs to transport goods from (x_i, y_i) to (m_i, n_i), its formula is as follows:

$$\mathrm{TC}_{\text{trans}}(t_i) = \left(\left|x_i - m_i\right| + \left|y_i - n_i\right|\right). \tag{3}$$

In the robot task scheduling model, in addition to the time consumption of autonomous robot executing tasks, there is also the time consumption between tasks. Assuming that the destination (x_i, y_i) of a robot executing task t_i and the starting point of the next task z_i to be executed by the robot is (m_i, n_i), the time consumption between two tasks is

$$C_{\text{between}}(t_i, z_i) = \left(\left|m_i - x_i\right| + \left|n_i - y_i\right|\right). \tag{4}$$

Therefore, the multirobot task assignment problem is transformed into two objective functions: maximizing the time consumption of a single robot and minimizing the total time for robots to perform tasks. Maximizing the time consumption of a single robot refers to the maximum time consumption MRC of a single robot among the robots completing the task, as shown in the following formula:

$$\begin{aligned} \text{MRC} &= \max \text{TC}_{\text{total}}(r_i), \\ \text{TC}_{\text{total}}(r_i, S_i) &= \sum_{j=1}^{k} \text{TC}\left(t_{i,j}\right) + \sum_{n=2}^{k} C_{\text{between}}\left(t_{i,n-1}, t_{i,n}\right), \end{aligned} \tag{5}$$

where TC_{total} denotes the total time consumed by the robot r_i to execute a task sequence S_i: $t_{i1} \longrightarrow t_{i2} \longrightarrow t_{i3} \cdots \longrightarrow t_{ik}$, t_{ik} denotes the i_{th} robot performing the k_{th} task. $C_{between}$ $(t_{i,n-1}, t_{i,n})$ represents the time consumed by the i_{th} robot when executing the task, k denotes the total number of tasks performed by the i_{th} robot, and n denotes the n_{th} task in execution.

Total robot time consumption minimization (MTC) is the second objective function; we need to minimize the total time for robots to perform tasks, and the formula is shown below:

$$\text{MTC} = \sum_{i=1}^{N} \text{TC}_{\text{total}}(r_i, S_i), \tag{6}$$

where N denotes the total number of robots in the intelligent warehouse.

All the above time consumption functions can be calculated using the Manhattan distance, since the speed of the robot in the warehouse is set to 1 m/s. The total distance traveled by the robots is also equal to the total time consumed by the robots.

2.2. Generation of Offspring. In the iterative optimization of discrete decision variables, some special crossover and mutation methods are used to form new offspring.

To solve the robot task assignment problem, we redesigned chromosome part and divided the chromosome into two parts. The first part is a set of task sequences, and the other part contains the number of tasks undertaken by each robot, and the sum of numbers is the total number of tasks in the task list. Figure 2 illustrates the relationship between the first part of the chromosome and the second part of the chromosome. As shown in Figure 2, robot 1 performs three tasks, robot 2 performs two tasks, 4 and 5, and robot 3 performs the remaining tasks.

For discrete decision variables, the conventional crossover and mutation operators [21] are not appropriate. Therefore, for this particular chromosome, two crossover operators are used. For these two parts of the chromosomes, we use order crossover [22] and simulated binary crossover.

Mutation operation can usually increase the random exploration ability of the algorithm, preventing algorithms from falling into local optimal. For discrete decision variables, we use slight mutation to replace polynomial mutation. The mutation process is shown in Figure 3. Firstly, we select a point from these points as the mutation point, then select a substring from the chromosome, and finally insert the substring behind the mutation point to form a new string of chromosomes.

3. The Proposed Method

In recent research, most of the literature studies tend to solve MOPs based on Pareto-optimal, allowing the algorithm to keep approaching the true Pareto optimal front through nondominated sorting and many improved Pareto-based methods [23]. In addition, some population diversity conservation mechanisms ensure that populations remain well distributed in the objective space after nondominated sorting.

3.1. Construction of Nondominated Solution Set. To reduce the high time complexity of constructing nondominated solution sets in NSGA, NSGAII proposed a new nondominated sorting method to select solutions for the new population. Given the good performance of NSGAII in dealing with MOPs, here we use this hierarchical nondominated solution set construction scheme to construct new populations.

In the nondominated sorting process, the nondominated individuals are first selected into the first stratum, then, the second stratum is the set of nondominant individuals obtained after removing the first stratum individuals from the population, and so on. In the final selection, individuals in the first stratum are considered first, and then individuals in the second stratum is considered until the new population size is satisfied. Specific details can be found in reference [17]. Algorithm 1 is the process of constructing the nondominated solution set.

3.2. Maximin Fitness Function. In the continuous development of multiobjective optimization, some fitness functions are used as indicators to evaluate individuals in the population, among which the maximin function is applied to multiobjective optimization with its own characteristics. The formula is as follows:

$$\text{fitness}^i = \max_{j \neq i}\left(\min_k\left(f_k(x_i) - f_k(x_j)\right)\right), \tag{7}$$

where k denotes objective from 1 to m, i represents the i_{th} individual, and j denotes any individual except i_{th} individual. And the properties of the maximin function are as follows [24, 25]:

(1) The maximin fitness can reflect the dominance between individuals. The maximin value greater than, equal to, or less than zero means that the individual is a dominated individual, the individual is weakly dominated, and individual is nondominated in the population, respectively.

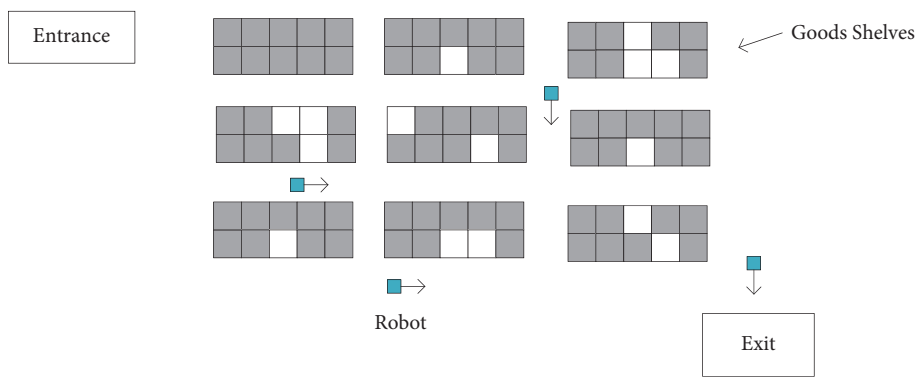


Figure 1: Floor plan of intelligent warehouse.

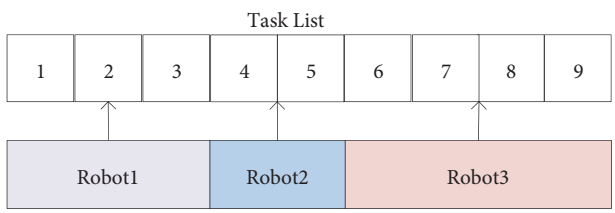


Figure 2: Example diagram of task assignment.

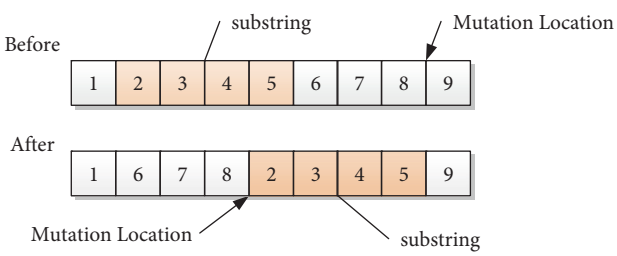


Figure 3: Example diagram of slight mutation.

(2) The maximin fitness reflects the presence or absence of clustering around an individual, as shown in Figure 4.

(3) The maximin function can be used to reflect the relationship between the dominated individual and the nondominated front. (see Figure 4).

3.3. One-by-One Comparison Strategy. The contribution of individual to the population can be effectively assessed by the maximin aggregation function. However, if only the maximin aggregation function is used to select suitable individuals, then the selection process may encounter the equivalence selection dilemma, that is, some individuals have equal maximin fitness, and there is no way to select a better individual from them, and then it is necessary to use a one-by-one comparison strategy to select a better individual from these individuals. The following figure shows an example of using the one-by-one comparison strategy to further select individuals [26].

Through one-by-one comparison strategy, the problem of equivalence selection can be solved efficiently. Suppose that three individuals are selected from the four individuals in Figure 5. First, the maximin aggregation function is used to evaluate these individuals. A with the minimum value is selected from these individuals to become the first candidate individual, then continue to select individuals from the remaining individuals through the maximin aggregation function, and the individual D is selected into the new population. Next, individuals B and C are compared with the new population by maximin function, and we select individual B to join the new population. Finally, the new population is formed by individuals A, B, D. As can be seen in the figure, the selected individuals remain well distributed.

3.4. Brain Storm Operator. Unlike most multiobjective algorithms, we use brain storm operator as a strategy for selecting parents, which increases the ability of the algorithm in terms of exploration and makes it less prone to fall into local optima. We demonstrate this part of the process with Algorithm 2.

3.5. Proposed Algorithm: MB-NSGA-II. The flow and framework of the algorithm are described below. In the first step, the population P is randomly initialized. And the brain storm operator is used to cluster parent population and then select appropriate chromosomes as parent chromosomes based on suitable conditions and generate the new population Q by crossover and mutation. After this, parent P and offspring Q are combined to form P'. Finally, the new population is regenerated by comprehensive selection. Keep cycling through the above steps until the *MAX_ Fitness Evaluations* is reached. The general framework of MB-NSGA-II is shown in Algorithm 4.

3.5.1. Normalization. In multiobjective optimization algorithms, normalization can effectively solve the problem that different objectives have widely varying ranges of values [27]. The formula is as follows:

$$f_m^{'}(x) = \frac{f_m(x) - z_m^{\text{lower}}}{z_m^{\text{upper}} - z_m^{\text{lower}}}, \tag{8}$$

where z_m^{lower} denotes the lower bound of the m_{th} objective function, z_m^{upper} denotes the upper bound of the m_{th} objective

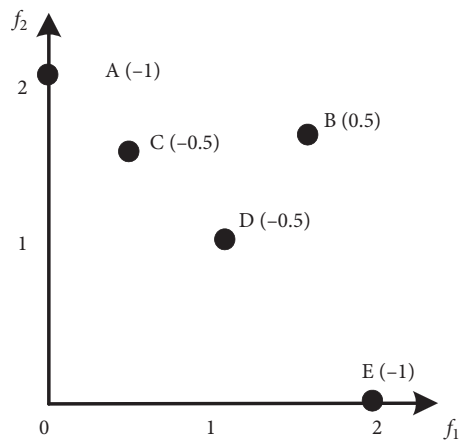


Figure 4: Properties of the maximin function.

Input: P (population)
Output: $P_1, P_2, \ldots, P_n$ (stratification results)
(1) $\forall p \in P, S_p = 0, D_p = \varnothing, i = 1$;// S_p denotes the size of set of solutions that dominate p, D_p i S denotes the set of solutions dominated by p
(2) **for** $\forall p \in P$
(3) **for** $\forall q \in P$
(4) if $(p \succ q)$ then $D_p = D_p \cup q$
(5) elseif $(q \succ p)$ then $S_p = S_p + 1$
(6) **end** for q
(7) if $(S_p = 0)$ then $P_1 = P_1 \cup p$
(8) **end** for p
(9) **while** $(P_i \neq \varnothing)$ // P_i indicates the number of non-dominated layers
(10) {POP= $\varnothing$;
(11) for $\forall p \in P_i$
(12) for $\forall q \in s_p$, $n_q = n_q - 1$;
(13) if $(n_q = 0)$ POP = POP $\cup q$
(14) end for p
(15) $i = i + 1$;
(16) P_i = POP ;
(17) **end** for while
(18) **end**
(19) **return** $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_\mathbf{n}$

Algorithm 1: Nondominated sorting.

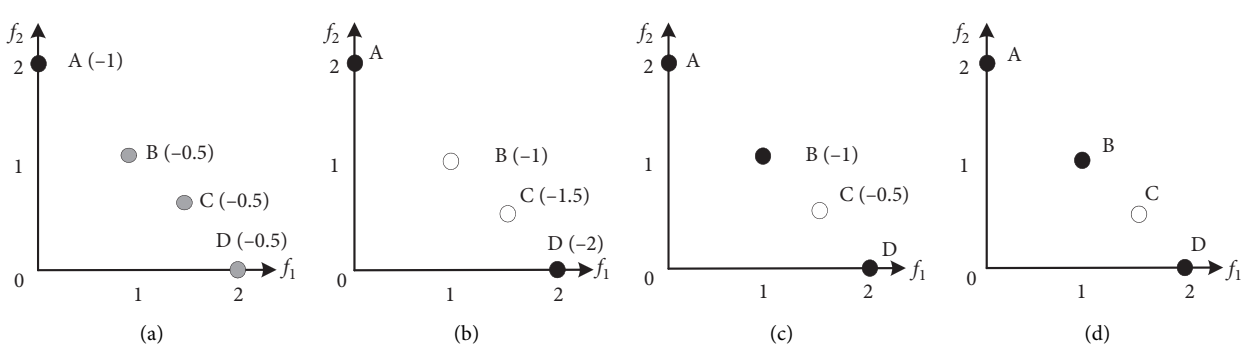


Figure 5: Diagram of the process of solving the equivalence selection problem by one-by-one comparison strategy.

```
Objective normalization (P)
for i = 1: S − 1
Clustering: Dividing the population into m clusters using the k-means method.
p_gen = rand(0, 1)
if p_gen < p_fixed
p_one = rand(0, 1) //Select a random cluster from m clusters
    if p_one < p_one-fixed//p_one-fixed is a predetermined value
Select the clustering center and the individual Pop_i as parent generation of population
    else
Select a random individual and the individual Pop_i as parent generation of population
    end
end
Compare the newly generated individuals with the present ones by maximin function, where the better one is preserved.
end
```

Algorithm 2: Brain storm operator (P, S).

function, m means this is the m_{th} objective function, and $m \in \{1, 2, \ldots, M\}$.

3.5.2. Comprehensive Selection. As shown in Algorithm 3, comprehensive selection consists of the following steps. First, nondominated sorting for the population *Pop* is executed to divide *Pop* into strata from 1 to n. If there is only one layer of nondominated layer, the maximin fitness value of the population and the ideal point is calculated, and individuals are selected into the new population one by one. Then, individuals were selected layer by layer starting from the first layer until the new population size is satisfied. When proceeding to stratum i, if the predetermined S is less than the number of individuals in the i_{th} stratum, the set of individuals P_s is selected using a one-by-one comparison strategy (Line 8 Algorithm 5), and finally P_s is merged with *Pop* to obtain the new population *Pop*.

3.5.3. One-by-One Comparison Strategy. The one-by-one comparison strategy is shown in Algorithm 5. One-by-one comparison strategy is used to compare individuals at the same stratum and select a subset of individuals from them. Firstly, let P_n be empty. First, each objective of the population P'' is normalized by (7).Then, if the population size of S is larger than *size*, comparing individual fitness values in S using the maximin aggregation function, select the individual with the smallest maximin fitness from it to join the *MF*, and so on, until P_n reaches the new population size.

3.6. Computational Complexity Analysis of MB-NSGA-II. To study the computational cost consumption of the algorithm, we summarize and discuss the time complexity of one generation in Algorithm 3. In addition to crossover and mutation operators, computational complexity is affected by the nondominated sorting and one-by-one comparison strategy.

First, normalization requires a computational complexity of $\text{O}(mN)$. And the nondominated sorting's computational complexity in Algorithm 1 is $\text{O}(mN^2)$. The complexity of calculating the maximin fitness value is $\text{O}(mN)$. The step of one-by-one comparison strategy needs $\text{O}(mN^2)$. In a word, the overall complexity of the algorithm is $O(mN^2)$ in one generation.

4. Experimental Study

The experimental part contains two sections; the first section discusses and analyzes the effectiveness of MB-NSGA-II on the benchmark test suites; the second part describes how we use our proposed algorithm to solve practical problems in intelligent warehouse.

4.1. Experimental Results on Benchmarks. This section provides the experimental results of MB-NSGAII with two other start-of-art multiobjective algorithms, NSGAII and IBEA, on the benchmark test suites DTLZ [28] and ZDT [29]. On DTLZ, each test instance was experimented on the 2,3,5- objectives, and on ZDT, the experiments were run on the 2-objective. Each algorithm needs to be run 20 times on each test issue. We can discuss the results of this experiment in terms of comprehensive performance indicator IGD.

4.1.1. Results and Discussion on ZDTs and DTLZs. The ZDT test suite contains six test problems with different characteristics, each test instance involving a feature that would cause the algorithm to have difficulty converging during optimization (e.g., multimodality). Through these test instances, it is possible to analyze which problems the algorithm excels in. DTLZ designs the problems through a systematic approach, and its decision variables and objectives are scalable, facilitating the algorithm to experiment in solving MaOPs. DTLZ increases the difficulty of the test problem by introducing manageable difficulties, facilitating the algorithm to experiment on test problems with any number of objectives. To more fully validate the effectiveness of several compared algorithms in solving MOPs, we chose to conduct experiments on 2,3, and 5 objectives. M in Table 1 indicates the objective number. The numbers of decision variable are set, as shown in Table 1. Details can be found in reference [28,29].

Input: *Off* (Offspring population), S (population size)
Output: *Pop* (new population)
$P_1, P_2, \ldots, P_n$ = **Nondominated sorting** (*Off*)
Pop = $\varnothing$;
if $n = 1$
Pop = **One-by-one comparison strategy** (Off, z^*, S);
else
while m
    Pop = Pop $\cup P_i$
    $i = i+1$
end for while
P_s = **One-by-one comparison strategy** (S_i, Pop, $S - |\text{Pop}|$) ;
Pop = $P'' \cup P_s$
end for if
return *Pop*

Algorithm 3: Comparehensive Selection (Off, S).

Initialization: Initial population P containing N randomly individuals
while *Fitness Evaluations* < *MAX_Fitness Evaluations* **do**
Q = **Brain Storm Operator** (P, s) ;
Off = $P \cup Q$
Pop = **Comprehensive _Selection** (*Off*, S);
end for while
return *Pop*

Algorithm 4: General Framework of MB-NSGA-II.

Input: S: individuals in stratum I, P'': new population, size: number of individuals needed for new population
Output: P_s: the set of individuals provided to the new population
$P_n = \varnothing$;
Objective normalization (Q);
if $|S|$>size
for $i = 1$: size-1
$MF = \text{argmin}_{i=\{1,2,\ldots,|S|\}}\{\text{fitness}^i\}$;//individual with minimal maximin fitness between individuals in S_i and the new population P
$P_n = P_n, \leq MF$;
end for
end if
Return $\mathbf{P_s}$

Algorithm 5: One-by-one comparison strategy ($S, P\prime\prime$, size).

To more visually reflect the ability of the algorithm on the benchmark test instances, we choose the comprehensive evaluation indicator HV and IGD as the basis for judging the performance of the algorithms. These two metrics are composed of different formulas, so the final population obtained by the algorithm can be evaluated from different perspectives. The formula and detailed description can be found in [30,31].

4.1.2. Experimental Settings

(i) Population size and termination conditions: we use population of size 100 for DTLZ with 2,3 objectives and 200 for DTLZ with 5 objectives. For ZDTs, we select population of size 100 for the three algorithms. For DTLZ1-4 and ZDT1-6, the algorithms will end after 500 generations.

(ii) Crossover and mutation: in this experiment, binary crossover and polynomial mutation were used as operators for crossover and mutation. Set 1.0 as the crossover probability and set $1/D$ as the probability of mutation, where D is the number of decision variables.

(iii) Selection of indicator: for IBEA, we select I_ε^+ as a performance indicator to evaluate the strengths and

Table 1: Setting of decision variables.

Problem	Number of decision variables
ZDT1-3	30
ZDT4,6	10
ZDT5	80
DTLZ1	$M+4$
DTLZ2-4	$M+9$

Table 2: Experimental results of IGD on ZDTs (mean and standard deviation).

Problem	M	NSGAII	IBEA	MBNSGAII
ZDT1	2	4.7879*e*-3 (1.31*e*-4) -	4.0901*e*-3 (1.01*e*-4) -	3.9627*e*-3 (6.92*e*-5)
ZDT2	2	4.9504*e*-3 (2.24*e*-4) +	8.4212*e*-3 (7.74*e*-4) -	5.3033*e*-3 (1.32*e*-4)
ZDT3	2	5.4102*e*-3 (1.31*e*-4) -	1.6020*e*-2 (8.05*e*-4) -	5.0404*e*-3 (1.24*e*-4)
ZDT4	2	5.0667*e*-3 (4.10*e*-4) +	2.2761*e*-2 (1.59*e*-2) -	6.3474*e*-3 (2.30*e*-4)
ZDT5	2	5.1086*e*-1 (8.31*e*-2) +	1.8513*e*+0 (4.39*e*-1)=	1.6593*e*+0 (7.13*e*-1)
ZDT6	2	3.7200*e*-3 (1.20*e*-4)=	4.4425*e*-3 (1.14*e*-4) -	3.7006*e*-3 (5.32*e*-5)

weaknesses of individuals. The details of the indicator can be seen in [32].

4.1.3. Analysis of Experimental Results. Comparison data of MB-NSGA-II, NSGA-II, and IBEA on DTLZ and ZDT are presented in Tables 2–5. These tables show the IGD and HV means and standard deviations of these compared algorithms on the benchmark test problems, and highlights the algorithm that worked best on these test problems. The Wilcoxon rank sum test with a significance level of 0.05 was used for the standard deviation analysis when conducting the experiment. The symbols "+," "−," and "=" are used to indicate that the algorithms are significantly better, significantly worse, or not significantly different between the compared algorithms and MB-NSGA-II.

In Table 2 and Table 3, MB-NSGA-II has an outstanding performance on both DTLZ test problems and ZDT test problems, and it ranks first on 11 out of 18 test instances, and it is also competitive on other test instances. In particular, it achieved the best results on DTLZ1 for both 2,3,5 objectives, and also ranked first on DTLZ3 for two objective problems. Of course, NSGA-II also had good results, obtaining the best results on the six problems. In Table 3, MB-NSGAII ranked first in performance on 8 test problems, while NSGAII values achieved good results on three problems. It can be seen that MB-NSGA-II achieved the best results on all test problems on DTLZ1 with hyperplane PF, which is a significant improvement over NSGAII. On DTLZ2, MB-NSGAII achieves the best results on 5 objectives, which also proves the effectiveness of maximin aggregation function in high-dimensional objective space. For DTLZ3, which is easily trapped in the local optimum and difficult to converge to the global optimum, MB-NSGAII achieves the best results on all three test problems except for the 2-objective. On DTLZ4, obtained after modifying the DTLZ2 function, MB-NSGAII achieves the best results on the 2-objective and 5-objective. Next two tables show the results of the algorithm under HV performance indicators. MB-NSGA-II has outstanding performance on most test instances. It can be seen from this that the validity of using the aggregation function to evaluate the populations. Of course, IBEA and NSGA-II also achieved the first in a few problems. The results in the table also prove that the strategy used in MB-NSGA-II is effective compared to NSGA-II.

4.2. Results and Discussion on Task Assignment Model. To confirm the ability of the algorithm to solve real-world problems, we built the robot assignment problem in the warehouse and solved it using the algorithm MB-NSGA-II. We complement the intelligent warehouse's situation mentioned above by first setting a two-dimensional coordinate system with the coordinates of the shipping gate of the warehouse as (0,100) and the coordinates of the incoming gate of the warehouse as (100,0). The intelligent warehouse has a certain number of autonomous robots in perfect condition and equal shipping speed of 1 m/s. All tasks to be performed in the warehouse are generated randomly. For a uniform comparison, we set the population of 500 for the compared algorithms and the termination condition to 200 generations.

To fully validate the performance of the algorithm in solving the task scheduling model, we assume that 5 or 10 robots are used to solve 100 randomly generated tasks; and 10 or 20 robots are used to solve 500 randomly generated tasks.

For real multiobjective optimization problems, we usually use the comprehensive performance indicator HV to measure the merit of the algorithm. To use HV as an indicator, the maximum values on each objective are selected as a reference points. In this real-world MOP, we select the maximum value that can be reached on each objective separately, and the specific details of HV can be found in [30]. Tables 6 and 7 show the HV values that the algorithm can obtain on 100 and 500 tasks, respectively. From the tables, the results show that MB-NSGAII obtains the maximum HV on all these tasks. That is, the populations

Table 3: Experimental results of IGD on several DTLZs (mean and standard deviation).

Problem	M	NSGAII	IBEA	MBNSGAII
DTLZ1	2	2.2068*e*-3 (7.35*e*-5) -	7.9965*e*-2 (7.99*e*-3) -	2.0230*e*-3 (5.45*e*-5)
	3	2.8447*e*-2 (1.33*e*-3) -	1.6616*e*-1 (2.51*e*-2) -	2.2376*e*-2 (3.68*e*-4)
	5	1.7146*e*-1 (1.25*e*-1) -	1.8372*e*-1 (2.16*e*-2) -	5.1833*e*-2 (3.37*e*-4)
DTLZ2	2	5.0483*e*-3 (1.75*e*-4) +	1.6671*e*-2 (1.60*e*-3) -	8.5238*e*-3 (2.92*e*-4)
	3	7.3058*e*-2 (2.57*e*-3) =	8.4162*e*-2 (2.51*e*-3) -	7.3973*e*-2 (1.57*e*-3)
	5	2.0563*e*-1 (4.67*e*-3) -	1.9251*e*-1 (1.54*e*-3) -	1.7906*e*-1 (2.44*e*-3)
DTLZ3	2	7.1268*e*-3 (1.39*e*-3) +	3.3913*e*-1 (8.73*e*-3) -	9.8870*e*-3 (1.10*e*-3)
	3	1.4200*e*-1 (2.06*e*-1) =	4.7891*e*-1 (4.51*e*-3) -	7.8834*e*-2 (2.64*e*-3)
	5	7.5535*e*-1 (7.31*e*-1) -	5.9365*e*-1 (9.76*e*-3) -	1.8301*e*-1 (2.58*e*-3)
DTLZ4	2	1.5241*e*-1 (3.11*e*-1) -	4.5174*e*-1 (3.75*e*-1) -	8.1849*e*-2 (2.32*e*-1)
	3	1.5811*e*-1 (2.77*e*-1) +	8.2918*e*-2 (2.68*e*-3) =	2.6157*e*-1 (2.41*e*-1)
	5	2.0630*e*-1 (3.41*e*-3) -	2.1267*e*-1 (7.27*e*-2) -	1.8232*e*-1 (1.93*e*-3)

Table 4: Experimental results of HV on several DTLZs (mean and standard deviation).

Problem	M	NSGAII	IBEA	MBNSGAII
DTLZ1	2	5.8121*e*-1 (3.60*e*-4) -	3.9969*e*-1 (1.87*e*-2) -	5.8162*e*-1 (3.86*e*-4)
	3	8.2082*e*-1 (5.02*e*-3) -	4.7837*e*-1 (5.50*e*-2) -	8.3698*e*-1 (1.86*e*-3)
	5	6.9302*e*-1 (3.39*e*-1) -	7.5172*e*-1 (4.51*e*-2) -	9.7599*e*-1 (1.40*e*-3)
DTLZ2	2	3.4654*e*-1 (1.97*e*-4) -	3.4613*e*-1 (2.17*e*-4) -	3.4731*e*-1 (1.09*e*-4)
	3	5.2551*e*-1 (3.99*e*-3) -	5.5477*e*-1 (1.48*e*-3) -	5.5689*e*-1 (9.61*e*-4)
	5	6.7753*e*-1 (7.73*e*-3) -	8.0904*e*-1 (1.36*e*-3) +	8.0024*e*-1 (2.49*e*-3)
DTLZ3	2	3.4146*e*-1 (2.73*e*-3) =	1.6879*e*-1 (3.86*e*-3) -	3.4251*e*-1 (2.27*e*-3)
	3	4.5747*e*-1 (1.56*e*-1) -	2.3645*e*-1 (9.09*e*-3) -	5.4725*e*-1 (3.20*e*-3)
	5	4.0235*e*-1 (3.46*e*-1) -	3.8077*e*-1 (6.39*e*-3) -	8.0179*e*-1 (4.10*e*-3)
DTLZ4	2	2.9554*e*-1 (1.08*e*-1) -	1.9305*e*-1 (1.32*e*-1) -	3.2164*e*-1 (8.11*e*-2)
	3	4.8619*e*-1 (1.39*e*-1) =	5.5574*e*-1 (9.75*e*-4) =	4.7330*e*-1 (1.09*e*-1)
	5	6.8256*e*-1 (8.53*e*-3) -	8.0150*e*-1 (2.60*e*-2) =	8.0780*e*-1 (1.73*e*-3)

Table 5: Experimental results of HV on ZDTs (mean and standard deviation).

Problem	M	NSGAII	IBEA	MBNSGAII
ZDT1	2	7.1925*e*-1 (1.76*e*-4) -	7.2017*e*-1 (1.36*e*-4) -	7.2031*e*-1 (8.51*e*-5)
ZDT2	2	4.4399*e*-1 (2.03*e*-4) -	4.4410*e*-1 (1.58*e*-4) -	4.4485*e*-1 (6.70*e*-5)
ZDT3	2	5.9938*e*-1 (8.34*e*-5) -	5.9816*e*-1 (1.17*e*-4) -	5.9973*e*-1 (4.53*e*-5)
ZDT4	2	7.1800*e*-1 (1.09*e*-3) =	7.0312*e*-1 (9.85*e*-3) -	7.1735*e*-1 (5.54*e*-4)
ZDT5	2	8.1676*e*-1 (1.20*e*-2) =	8.1119*e*-1 (9.16*e*-3)	8.1451*e*-1 (3.26*e*-4)
ZDT6	2	3.8826*e*-1 (1.18*e*-4) =	3.8766*e*-1 (1.17*e*-4) -	3.8832*e*-1 (5.60*e*-5)

Table 6: Experimental results of HV when robots perform 100 tasks.

Number of robots	IBEA	NSGA-II	MB-NSGAII
5	0.1379	0.1426	0.1445
10	0.2514	0.2341	0.2549

Table 7: Experimental results of HV when robots perform 500 tasks.

Number of robots	IBEA	NSGA-II	MB-NSGAII
10	0.1388	0.1374	0.1452
20	0.1805	0.1804	0.1853

Table 8: Average time consumption of robots when using 5 robots for 100 tasks.

	IBEA	NSGA-II	MB-NSGAII
Robot 1	3429.3	3358.3	3424.3
Robot 2	3589.9	3705	3719.2
Robot 3	3524.4	3520.3	3418.6
Robot 4	3424.3	3103.9	3031.7
Robot 5	3316.4	3301.3	3128

Table 9: Average time consumption of robots when using 10 robots for 500 tasks.

	IBEA	NSGA-II	MB-NSGAII
Robot 1	8894.5	9296.3	8227.2
Robot 2	8677.2	8882.1	8693.7
Robot 3	8516.4	7974.5	8681.3
Robot 4	8094.8	7616.1	7594.8
Robot 5	9015.4	8169	8170.3
Robot 6	7878	7454.1	7684.7
Robot 7	8511.7	7680.7	7364.1
Robot 8	8244.3	8189.8	9170
Robot 9	8665.4	9522.6	9233.9
Robot 10	8739.7	8210	8620.1

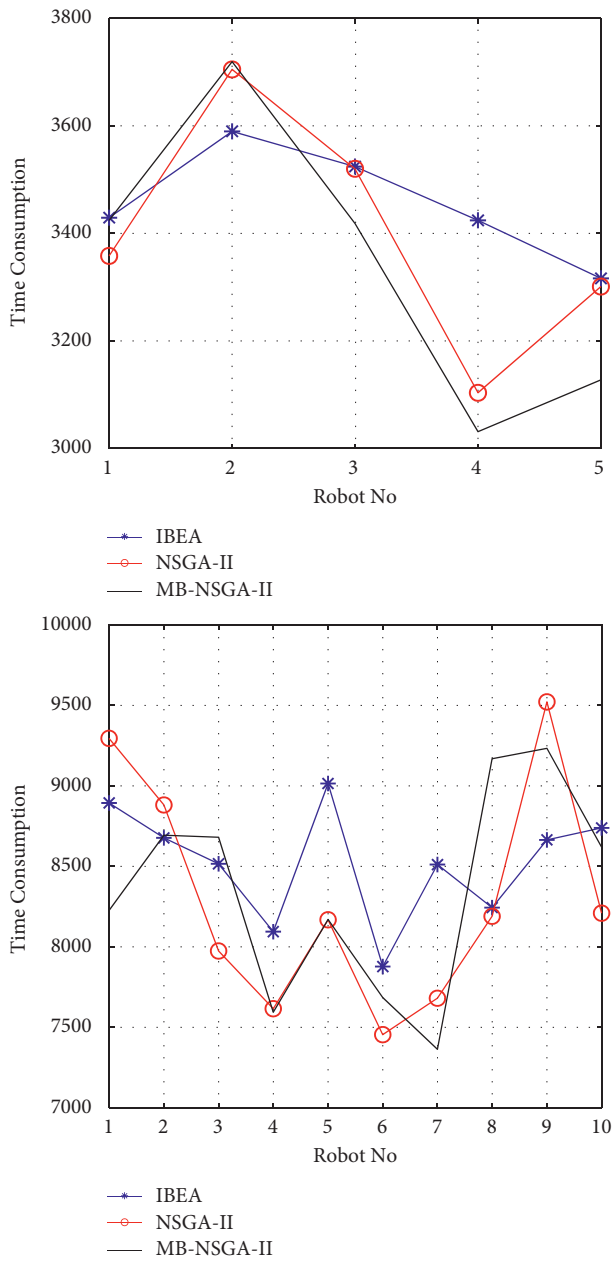


Figure 6: Schematic diagram of the average time consumed by robots to perform tasks.

obtained by MB-NSGAII have better convergence and distribution. Through the above experiments, it can be proved that MB-NSGAII can give managers better solutions.

Tables 8 and 9 show the average time consumption of robots when solving 100 tasks with 5 robots and 500 tasks with 10 robots, respectively. Since each robot runs at a speed of 1 m/s, their average running distance can also be represented by Tables 8 and 9. As can be seen from the tables, the time consumption of each robot is close to each other and load balancing is achieved. Figure 6 shows the average time consumption of the robot when performing 100 tasks and 500 tasks.

5. Conclusion

In this paper, we have proposed a corresponding mathematical model for robot task assignment problem in an intelligent warehouse and solve it by MB-NSGAII, which uses nondominated sorting, maximin aggregation function, and brain storm operator. MB-NSGA-II is validated on the DTLZ and ZDT test suites, and it achieves satisfactory results in solving the real-world MOP with robot task assignment.

Next, we will continue to investigate the use of maximin aggregation function and brain storm operator to solve MOPs. We will also use this approach to solve more real-world MOPs.

Data Availability

The data are not available due to the nature of this research as participants of this study did not agree for their data to be shared publicly.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by Liaoning Social Science Planning Fund Key Project (L20CGL012); Liaoning Science and Technology Department Science Career Public Welfare Research Fund (2021JH4/10100026); the Fundamental Research Funds for the Central Universities No. N2117005; the Joint Funds of the Natural Science Foundation of Liaoning Province und Grant 2021-KF-11-01. It is noted that, unlike earlier work [33], this paper provides some improvements: a new 2-objective mathematical model, i.e., minimum total time consumed by all robots and maximum time consumed by a single robot, for the robot task scheduling problem, and then propose a novel nondominated sorting algorithm to solve it.

References

[1] X. Yao, C. Yuanyuan, Z. Li, and S. Malin, "Green efficiency performance analysis of the logistics industry in China: based on a kind of machine learning methods," *Annals of Operations Research*, vol. 308, no. 1, pp. 727–752, 2022.

[2] H. Qin, J. Xiao, D. Ge et al., "JD.com: operations research algorithms drive intelligent warehouse robots to work," *INFORMS Journal on Applied Analytics*, vol. 52, no. 1, pp. 42–55, 2022.

[3] M. Petrović, M. Zoran, and J. Aleksandar, "A novel methodology for optimal single mobile robot scheduling using whale optimization algorithm," *Applied Soft Computing*, vol. 81, 2019.

[4] M. G. C. A. Cimino, D. Minici, M. Monaco, S. Petrocchi, and G. Vaglini, "A hyper-heuristic methodology for coordinating swarms of robots in target search," *Computers & Electrical Engineering*, vol. 95, Article ID 107420, 2021.

[5] W. Chi, Z. Ding, J. Wang, G. Chen, and L. Sun, "A generalized voronoi diagram-based efficient heuristic path planning method for RRTs in mobile robots," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 5, pp. 4926–4937, 2021.

[6] S. Mosallaeipour, M. G. Nejad, S. M. Shavarani, and R. Nazerian, "Mobile robot scheduling for cycle time optimization in flow-shop cells, a case study," *Production Engineering*, vol. 12, no. 1, pp. 83–94, 2018.

[7] F. Gul, W. Rahiman, S. S. N. Alhady, A. Ali, I. Mir, and A. Jalil, "Meta-heuristic approach for solving multi-objective path planning for autonomous guided robot using PSO-GWO optimization algorithm with evolutionary programming," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 7, pp. 7873–7890, 2021.

[8] Y. Liu, B. Y. Ooi, and D. Qin, "Dynamic order-based scheduling algorithms for automated retrieval system in smart warehouses," *IEEE Access*, vol. 9, pp. 158340–158352, 2021.

[9] Z. Tan, H. Li, and X. He, "Optimizing parcel sorting process of vertical sorting system in e-commerce warehouse," *Advanced Engineering Informatics*, vol. 48, Article ID 101279, 2021.

[10] D. Guo, Z. Lyu, W. Wu, R. Y. Zhong, Y. Rong, and G. Q. Huang, "Synchronization of production and delivery with time windows in fixed-position assembly islands under Graduation Intelligent Manufacturing System," *Robotics and Computer-Integrated Manufacturing*, vol. 73, Article ID 102236, 2022.

[11] L. Ma, X. Wang, X. Wang, L. Wang, Y. Shi, and M. Huang, "TCDA: truthful combinatorial double auctions for mobile edge computing in industrial internet of things," *IEEE Transactions on Mobile Computing*, p. 1, 2021.

[12] X. Wu and M. Zhang, "An intelligent algorithm for AGV scheduling in intelligent warehouses," *Lecture Notes in Computer Science*, vol. 12689, pp. 163–173, 2021.

[13] L. Ma, M. Huang, S. Yang, R. Wang, and X. Wang, "An adaptive localized decision variable analysis approach to large-scale multiobjective and many-objective optimization," *IEEE Transactions on Cybernetics*, 2021.

[14] H. Ishibuchi, T. Noritaka, and N. Yusuke, "Evolutionary manyobjective optimization: a short review," in *Proceedings of the CEC 2008*, pp. 2419–2426, IEEE, Hong Kong, China, 1 Jun. 2008.

[15] L. Ma, N. Li, Y. Guo et al., "Learning to optimize: reference vector reinforcement learning adaption to constrained many-objective optimization of industrial copper burdening system," *IEEE Transactions on Cybernetics*, 2021.

[16] L. S. Batista, F. Campelo, F. G. Guimarães, and J. A. Ramírez, "A comparison of dominance criteria in many-objective optimization problems," in *Proceedings of the CEC 2011*, pp. 2359–2366, IEEE, New Orleans, LA, U.S.A, 5 June 2011.

[17] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Proceedings of the Parallel*

Problem Solving from Nature PPSN VI, pp. 849–858, Springer, Paris, France, 18 January 2000.

[18] M. Emmerich, N. Beume, and B. Naujoks, "An EMO algorithm using the hypervolume measure as selection criterion," *Lecture Notes in Computer Science 2005*, Springer, pp. 62–76, Berlin, Heidelberg.

[19] Y. Shi, "Brain storm optimization algorithm," *Lecture Notes in Computer Science*, Springer, pp. 303–309, Berlin, Heidelberg.

[20] H. Zhang, Z. Guo, W. Zhang et al., "Layout design for intelligent warehouse by evolution with fitness approximation," *IEEE Access*, vol. 7, pp. 166310–166317, 2019.

[21] K. Deb, K. Sindhya, and T. Okabe, "Self-adaptive simulated binary crossover for real-parameter optimization," in *Proceedings of the GECCO 2007*, pp. 1187–1194, ACM, London, England, U.K, 7 July 2007.

[22] L. Davis, "Applying adaptive algorithms to epistatic domains," in *Proceedings of the IJCAI 1985*, pp. 162–164, ACM, San Francisco, CA, U.S.A, 2 August 1985.

[23] L. Ma, S. Cheng, and Y. Shi, "Enhancing learning efficiency of brain storm optimization via orthogonal learning design," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 11, pp. 6723–6742, 2021.

[24] E. J. P. Solteiro, P. B. D. M. Oliveira, and J. A. T. Machado, "Multi-objective MaxiMin sorting scheme," *Lecture Notes in Computer Science 2005*, Springer, pp. 165–175, Berlin, Heidelberg.

[25] X. Li, "Better spread and convergence: particle swarm multiobjective optimization using the maximin fitness function," *Genetic and Evolutionary Computation - GECCO 2004 2004*, Springer, pp. 117–128, Berlin, Heidelberg.

[26] Y. Liu and D. J. Y. Gong, "A many-objective evolutionary algorithm using a one-by-one selection strategy," *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2689–2702, 2017.

[27] Q. He, X. Wang, Z. Lei, M. Huang, Y. Cai, and L. Ma, "TIFIM: a two-stage iterative framework for influence maximization in social networks," *Applied Mathematics and Computation*, vol. 354, pp. 338–352, 2019.

[28] K. Deb, T. Lothar, L. Marco, and Z. Eckart, "Scalable test problems for evolutionary multi-objective optimization," *Evol. Multi. Opti, Theoretical Advances and Applications*, Springer, pp. 105–145, London, U.K.

[29] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[30] L. While, P. Hingston, L. Barone, and S. Huband, "A faster algorithm for calculating hypervolume," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 29–38, 2006.

[31] C. A. C. Coello and Corte, "Solving multiobjective optimization problems using an artificial immune system," *Genetic Programming and Evolvable Machines*, vol. 6, no. 2, pp. 163–190, 2005.

[32] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," *PPSN VIII Lecture Notes in Computer Science*, Springer, pp. 832–842, Berlin, Germany.

[33] S. Yang, Y. Zhang, L. Ma et al., "A novel maximin-based multi-objective evolutionary algorithm using one-by-one update scheme for multi-robot scheduling optimization," *IEEE Access*, vol. 9, pp. 121316–121328, 2021.

Hindawi
Discrete Dynamics in Nature and Society
Volume 2022, Article ID 5291296, 10 pages
https://doi.org/10.1155/2022/5291296

Research Article

A Hybrid Particle Swarm Optimizer for Curriculum Sequencing Problem

Xianjie Peng,[1] Xiaonan Sun,[2] and Zhen He[3]

[1]*Department of Human Resources, Tongji University, Shanghai, China*
[2]*Institute of Vocational Education, Tongji University, Shanghai, China*
[3]*Faculty of Education, Beijing Normal University, Beijing, China*

Correspondence should be addressed to Zhen He; zhenhemail163@163.com

Received 20 March 2022; Accepted 11 May 2022; Published 6 June 2022

Academic Editor: Shi Cheng

Curriculum sequencing problem is crucial to e-learning system, which is a NP-hard optimization problem and commonly solved by swarm intelligence. As a form of swarm intelligence, particle swarm optimization (PSO) is widely used in various kinds of optimization problems. However, PSO is found ineffective in complex optimization problems. The main reason is that PSO is ineffective in diversity preservation, leading to high risks to be trapped by the local optima. To solve this problem, a novel hybrid PSO algorithm is proposed in this study. First, a competitive-genetic crossover strategy is proposed for PSO to balance the convergence and diversity. Second, an adaptive polynomial mutation is introduced in PSO to further improve its diversity preservation ability. Furthermore, a curriculum scheduling model is proposed, where several constraints are taken into considerations to ensure the practicability of the curriculum sequencing. The numerical comparison experiments show that the proposed algorithm is effective in solving function optimization in comparison to several popular PSO variants; furthermore, for the optimization of the designed curriculum sequencing problem, the proposed algorithm shows significant advantages over the compared algorithms with respect to the degree of the satisfaction of the objectives, i.e., 20, 14, and 5 percentages higher, respectively.

1. Introduction

With the rapid developments of communication technologies, e-learning is growing as an important teaching method. However, the traditional e-learning systems commonly ignore reusability and are not adaptive [1]. Furthermore, the traditional e-learning systems are inefficient in flexibility and cannot provide either the adaptive learning or the fixed curriculum path for students [2, 3]. Due to its characteristic of NP hard, individualizing paths for each student are costly to providers, since teachers typically perform the sorting process manually. Therefore, it is of great value to automate the curriculum sequencing to reduce the cost of high-quality e-learning. To solve this issue, many kinds of models and optimization methods are carried out for curriculum sequencing problems [4–11], where particle swarm optimization (PSO) [12] is widely adopted as the optimizer.

As a form of swarm intelligence, PSO builds on the population-based searching technique, where each particle (A feasible solution in the swarm) includes two attributes, i.e., the position and the velocity. The former represents the information of decision variables of the particles and the latter is the evolutionary direction of the particles. Based on equations (1) and (2), the swarm is able to keep updating and traverse the decision space to find better solutions:

$$v_i^d(t+1) = v_i^d(t) + r_1 c_1\left(p_i^d(t) - \text{gbest}^d\right) + r_2 c_2\left(p_i^d(t) - \text{pbest}_i^d\right), \tag{1}$$

$$p_i^d(t+1) = p_i^d(t) + v_i^d(t+1), \tag{2}$$

where $v_i^d(t)$ is the dth dimension of the position of the ith particle in generation t, gbest is the best particle of the current swarm and pbest_i is the best position searched by the ith particle so far, v_i is the velocity of the ith particle, and c_1 and c_2 are the coefficients, while r_1 and r_2 are two random number generated within $(0, 1)$. As one can see, PSO is of high simplicity.

However, PSO is found ineffective in complex problems, such as the multimodal and large-scale optimization problems [13, 14]. The main reason is that PSO lacks global searching ability and is prone to be trapped by the local optimum due to its poor ability in balancing the exploration and exploitation.

To this end, a huge amount of effort has been carried out, mainly being classified into three categories: the improvements on parameters [15–19], different kinds of topologies [20–22], and hybridizations with other kinds of techniques and algorithms [23, 24]. Nevertheless, the existing algorithms still cannot obtain the global optima when solving complex problems. The main reason is that the current PSO and its variants are still inefficient in diversity preservation. For instance, CSO tries to enhance its diversity preservation ability by guiding the updated particles with the mean position of the whole swarm. However, the mean position is shared by all the updated particles, resulting in risks in premature convergence.

To improve the diversity preservation ability of PSO, this study proposes a novel hybrid particle swarm optimization algorithm; the main contributions of this study are listed as follows:

(1) A competitive-genetic crossover operation is proposed, which combines the competitive mechanism, the genetic crossover, and the elite strategy to improve the diversity of the exemplars and ensure a comprehensive exploitation ability of PSO

(2) An adaptive polynomial mutation is introduced in PSO to further enhance its diversity preservation ability, leading to the proposed hybrid particle swarm optimization algorithm

(3) A curriculum sequencing model is proposed and the proposed algorithm is applied into solving the curriculum sequencing problem to improve the efficiency of the e-learning systems

The rest of this study is organized as follows. Section 2 presents the literature review for the mainstream of the improvements on PSO. Section 3 provides a detailed description of the proposed algorithm. The experiments and the case study are conducted in Section 4, and we conclude this study in Section 5.

2. Related Work

As mentioned above, the current works for improving PSO can be mainly categorized into the following three classes.

2.1. Control of the Parameters. Different parameters in PSO are of different functions: the parameter in inertia component mainly focuses on keeping the particles move according to the historical information for global optimization; the parameters in the social components are related to the convergence. Shi and Eberhart propose a linear control method to dynamically adjust the inertia weight [25]. The main goal of their work is to enhance the diversity preservation ability for PSO in the early optimization stage and emphasize the convergence ability for PSO in the later stage. Furthermore, they propose a fuzzy adjustment strategy for the inertia weight [26]. Ratnaweera et al. propose HPSO-TVAC to dynamically adjust the acceleration coefficients during the run [16]. Zhan et al. propose the adaptive particle swarm optimization algorithm, changing the acceleration coefficients according to a predesigned evolutionary state estimation strategy [18]. Piotrowski et al. put forward a study to investigate the influences of the swarm size on the performance of PSO [27]. Tian and Shi propose a PSO variant, where a logistic map and a sigmoid-like inertia weight are utilized to initialize the swarm and balance the exploration and exploitation, respectively [28].

2.2. Modifications on the Topologies. The main idea of the methods in this category is designing new information exchanging topologies to improve the diversity preservation ability to enhance the search ability of PSO. Mendes et al. investigate a fully informed topology, where particles learn from their neighbors instead of the global best particle in the swarm [20]. Liang et al. propose comprehensive learning strategy which allows each particle's historical personal best position to be the exemplar for others [21]. Chen et al. propose ALCPSO, which dynamically replaces the global best particle with another individual according to several predefined rules to guide the updated particles [29]. Cheng et al. propose FBE, where the whole swarm is divided into two subswarms, and a competitive mechanism is designed to select exemplars for particles [30]. The weak particles in the competition learn from the best particle in the internal subswarm and randomly selected particles from the external subswarm, while a mutation operation is executed on the strong particles. Furthermore, Cheng and Jin propose the competitive swarm optimizer (CSO), which only updates a half of the particles and adopts the strong particles in the competition to guide the weak particles [22]. Inspired by the social learning actions, Cheng and Jin propose SLPSO, which allows particles to learn from all the particles that are better than themselves [31]. Yang et al. propose DLLSO [32], which divides particles into different levels based on their fitness value. Afterwards, each particle chooses two different exemplars from superior levels. Zhang et al. propose a modified PSO by introducing a dynamic neighborhood-based learning strategy to enhance the diversity preservation ability for PSO [33]. Zeng et al. propose DNSPSO, where a novel velocity updating mechanism is designed to adjust the personal best position and the global best position based on a designed distance-based dynamic neighborhood to enhance the information sharing among the particles [34].

2.3. Hybridization with Other Techniques. Hybridizing PSO with other techniques is able to make use of the advantages of both PSO and other techniques. Van et al. propose CCPSO-SK and CCPSO-HK by integrating the co-operative co-evolutionary framework and PSO to solve the large-scale optimization [35, 36]. Li and Yao put forward CCPSO2, where the Gaussian and Cauchy mutation are proposed to update individuals to balance the diversity and convergence [37]. Qin et al. propose a PSO variant by dividing the whole swarm into learned and learning subswarms at each generation; afterwards, the learning subswarm will learn from the learned subswarm with respect to a random probability [38]. Li et al. propose a hybrid algorithm by incorporating the update mechanism of PSO into biogeographic optimization algorithm to improve the exploration ability of the algorithm [12]. The genetic learning particle swarm optimization put forward by Gong et al. adopts crossover and mutation operators to enhance the exploration ability of PSO [39]. Similarly, Chen et al. design two kinds of crossover operations to breed promising exemplars [23]. Chen et al. propose HPSOSSM, which uses a logistic map sequence to enhance the diversity and adopts a spiral-shaped mechanism to improve the convergence speed [40].

3. Proposed Algorithm

As discussed above, the main issue in PSO is that PSO is ineffective in diversity preservation. Targeting against this issue, this study puts forward two improvements for PSO: a competitive-genetic crossover operator and an adaptive mutation operator. The details are presented as follows.

3.1. Competitive-Genetic Crossover Operator. Competitive mechanism-based learning strategy proposed in [22] has been demonstrated to be effective in both convergence and diversity preservation. The reason is that it not only can be used to select exemplars for particles for convergence but also is beneficial to the diversity preservation by only updating half of the particles. On the contrary, genetic algorithm (GA) [41] potentially has better convergence compared with PSO, since the promising information in the population of GA can be directly copied to other solutions and kept to the offsprings, while PSO updates its particles according to stochastic strategies. Motived by this, a competitive-genetic crossover operator (CGCO) is proposed to conduct convergence and improve the diversity preservation ability for PSO.

First, the competitive mechanism is conducted to determine the strong particles' set (the winners in the competition) and the weak particles' set (the losers in the competition) at each generation, which can be shown in Figure 1. To be specific, (i) the whole swarm is randomly divided into $N/2$ subswarms, i.e., each subswarm includes two particles, (ii) the particles in each subswarm conduct the competition according to their fitness to determine the strong particle and the weak particle, i.e., the winner and loser in the competition, respectively.

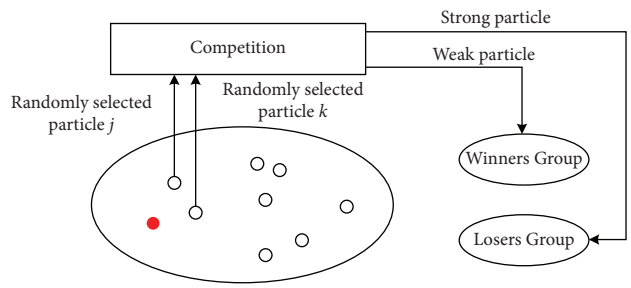


Figure 1: The competition mechanism.

Second, a multipoint crossover is conducted on the paired strong particles and the corresponding weak particles. Different from the mechanism in GA, only the strong particles' information will be copied into the weak particles in the proposed crossover operator. On the contrary, pbest of the particles are further adopted in the crossover operation. The crossover operation for a particle $p_{l,i}$ can be formulated as

$$\begin{aligned} p_{l,i}[:\,] &= \text{Concatenat}\left(p_{l,i}[0{:}\ \text{pos}_1], p_{w,i}[\text{pos}_1{:}\ \text{pos}_2], p_{l,i}[\text{pos}_2{:}\ ]\right), \\ p_{l,i}[:\,] &= \text{Concatenat}\left(p_{l,i}[0{:}\ \text{pos}_3], \text{pbest}_i[\text{pos}_3{:}\ \text{pos}_4], p_{l,i}[\text{pos}_4{:}\ ]\right), \end{aligned} \tag{3}$$

where $p_{l,i}[:\,]$ is the position of the ith particle in the losers group, $p_{w,i}$ is the corresponding winner and pbest is the historical personal best position of $p_{l,i}$, and pos_1, pos_2, pos_3, and pos_4 are four randomly generated indexes for crossover.

Finally, an elite strategy is adopted: an offspring is kept to the next generation if it has improvements in comparison to its parent (the corresponding losers) or the offspring will be kept if a generated random number is less than 0.5.

Together, the pseudocode of CGCO is shown in Algorithm 1.

3.2. Adaptive Mutation Operator. Although CGCO is beneficial to diversity by only updating half of the particles, it only exchanges existing information between the particles. Therefore, there still exist high risks that the swarm will be trapped by the local optima. To solve this problem, an adaptive mutation operator (AMO) is proposed.

First, each particle is initialized with a counter in the swarm initialization stage.

Second, in the optimization stage, the mutation rate for the ith particle is computed according to

$$Mu_r(i) = Mu_{\max}\frac{\text{counter}(i)}{g_n}, \tag{4}$$

where counter(i) records how many times that the ith particle successfully has been updated, while g_n is the cost number of the generation and $Mu_{\max}$ is the predefined maximum of the mutation rate.

With Mu_r, each particle conducts the polynomial mutation to improve the diversity of the swarm. One can find that the better a particle, the smaller the corresponding mutation rate. This is reasonable since the information of the

```
Input: swarm P(t), fitness value vector fitness, and swarm size N
Output: The offsprings P_off
P_off ← ∅
P_winners and P_losers ← Conduct the competition mechanism
for i = 1 to N/2 do
    p_{l,i}, p_{w,i} ← Select the ith particle and the corresponding winner in P_losers and P_winners
    p_1 ← Conduct the crossover operation between p_{l,i} and p_{w,i}
    if p_off is better than p_{l,i} then
        P_off ← P_off ∪ p_off
    else
        if rand ≤ 0.5 then
            P_off ← P_off ∪ p_{l,i}
        else
            P_off ← P_off ∪ p_off
        end
    end
end
```

Algorithm 1: Pseudocode of CGCO.

promising particles should be kept with a large likelihood for convergence.

3.3. HPSO-GA. To sum up, a competitive-genetic crossover operator and an adaptive mutation strategy are proposed to conduct convergence and diversity preservation for PSO. By integrating these two strategies together, a novel hybrid PSO is proposed, which is refer to HPSO-GA in the following for simplicity. The procedure of HPSO-GA can be found in Figure 2.

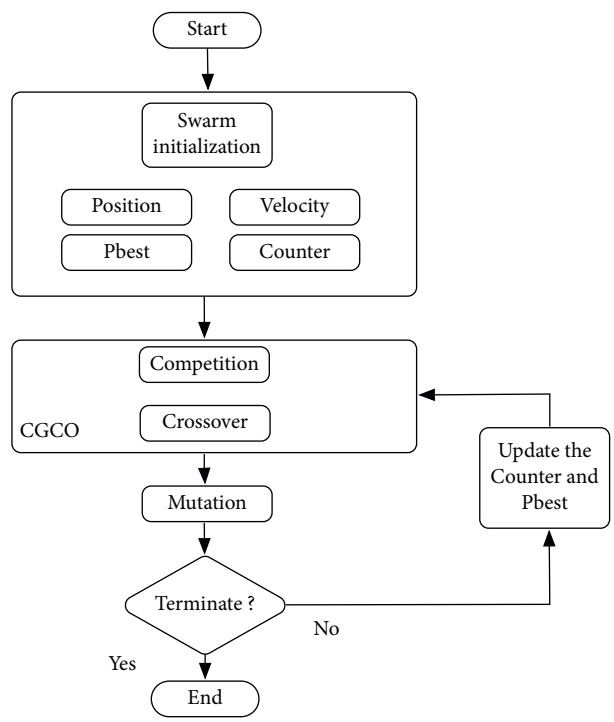


Figure 2: The flowchart of the proposed algorithm.

4. Experiments and Results

In this section, numerical comparisons are first conducted to test the performance of the proposed HPSO-GA; second, a curriculum sequencing model is put forward. Afterwards, HPSO-GA and other three algorithms are applied into solving the proposed curriculum sequencing problem to test the reliability of the proposed algorithm.

4.1. Numerical Comparisons

4.1.1. Compared Algorithms. To test the numerical optimization performance of HPSO-GA, five popular PSO variants are selected in the comparisons, including ALCPSO [29], LIPS [20], CSO [22], HPSO-TVAC [16], and DLLSO [32]; the benchmarks posted by CEC 2013 is adopted to test the algorithms.

4.1.2. Experimental Settings. For a fair comparison, all the compared algorithms adopt the suggested parameter settings in the corresponding references; the swarm size for the compared algorithms and the dimensionality of the benchmarks are set to 100; the maximum number of the fitness evaluations is adopted as the termination criterion, which is set to $5E+05$. The parameter settings for HPSO-GA are set as follows: the crossover rate is set to 1 and the maximum mutation rate is set to 0.05; each algorithm is run 31 times on each benchmark. The Wilcoxon rank sum test is adopted for the statistical analysis between the peer algorithms and HPSO-GA, where the significance level is set to 0.05.

4.1.3. Results. Table 1 shows the results of the comparison, where the mean of the corresponding results are recorded. The symbols "+," "−," and "=" in the bottom of the table

Table 1: The comparison on CEC 2013 benchmarks ($D = 100$).

Functions	Property	ALCPSO	CSO	HPSO-TVAC	LIPS	DLLSO	HPSO-GA
F_1	Mean	7.47E-12	2.28E-13	2.72E-08	8.71E-01	6.37E-13	2.20E-13
	p-value	**5.25E-10**$^+$	**5.00E-03**$^+$	**5.35E-10**$^+$	**5.35E-10**$^+$	**1.38E-08**$^+$	-
F_2	Mean	1.32E+08	3.17E+06	2.65E+07	4.09E+08	1.76E+07	3.05E+06
	p-value	**1.42E-09**$^+$	5.61E-01$^=$	**1.42E-09**$^+$	**1.42E-09**$^+$	**2.56E-09**$^+$	-
F_3	Mean	4.64E+10	1.33E+08	2.16E+10	3.24E+13	9.15E+09	3.27E+08
	p-value	**1.41E-09**$^+$	6.75E-06-	**1.41E-09**$^+$	**1.41E-09**$^+$	**1.42E-09**$^+$	-
F_4	Mean	5.20E+04	8.76E+04	6.66E+04	2.52E+05	7.35E+04	2.38E+04
	p-value	**2.02E-09**$^+$	**1.41E-09**$^+$	**1.41E-09**$^+$	**1.41E-09**$^+$	**1.60E-09**$^+$	-
F_5	Mean	3.02E-12	1.14E-13	2.77E-04	1.45E+03	1.03E-05	3.22E-13
	p-value	**7.18E-10**$^+$	4.16E-11-	**7.40E-10**$^+$	**7.40E-10**$^+$	**7.25E-10**$^+$	-
F_6	Mean	6.83E+02	2.00E+02	2.40E+02	2.95E+03	2.73E+02	1.97E+02
	p-value	**1.42E-09**$^+$	**1.41E-09**$^+$	**1.42E-09**$^+$	**1.42E-09**$^+$	**1.42E-09**$^+$	-
F_7	Mean	1.80E+02	1.10E+01	5.17E+04	2.66E+03	6.90E+01	2.00E+00
	p-value	**1.42E-09**$^+$	**1.41E-09**$^+$	**1.42E-09**$^+$	**1.42E-09**$^+$	**1.60E-09**$^+$	-
F_8	Mean	2.20E+01	2.20E+01	2.20E+01	2.20E+01	2.10E+01	2.05E+01
	p-value	8.77E-01$^+$	**5.01E-05**$^+$	**1.30E-02**$^+$	**1.17E-02**$^+$	5.35E-01$^=$	-
F_9	Mean	1.26E+02	4.20E+01	1.37E+02	1.18E+02	5.90E+01	2.56E+01
	p-value	**1.42E-09**$^+$	**1.60E-03**$^+$	**1.42E-09**$^+$	**1.42E-09**$^+$	**2.21E-07**$^+$	-
F_{10}	Mean	1.20E+01	1.64E-01	2.60E+01	1.37E+03	2.50E+01	1.21E-01
	p-value	**1.42E-09**$^+$	4.15E-01$^=$	**1.42E-09**$^+$	**1.42E-09**$^+$	**1.42E-09**$^+$	-
F_{11}	Mean	2.57E+02	5.20E+01	3.22E+02	7.45E+02	1.18E+02	4.40E+01
	p-value	**1.41E-09**$^+$	2.80E-06$^+$	**1.41E-09**$^+$	**1.41E-09**$^+$	**8.28E-09**$^+$	-
F_{12}	Mean	6.70E+02	8.04E+02	1.72E+03	7.99E+02	1.40E+02	7.99E+02
	p-value	3.31E-04-	**9.90E-01**$^=$	**1.41E-09**$^+$	7.415E-01$^=$	1.41E-09-	-
F_{13}	Mean	1.05E+03	8.01E+02	2.11E+03	1.30E+03	3.75E+02	7.95E+02
	p-value	**7.38E-09**$^+$	**3.00E-03**$^+$	**1.42E-09**$^+$	**1.42E-09**$^+$	1.42E-09-	-
F_{14}	Mean	6.58E+03	1.45E+03	3.77E+03	1.11E+04	5.19E+03	2.33E+03
	p-value	**1.42E-09**$^+$	4.67E-06-	**1.31E-07**$^+$	**1.42E-09**$^+$	**2.03E-09**$^+$	-
F_{15}	Mean	2.39E+04	2.92E+04	2.11E+04	1.41E+04	9.30E+03	2.11E+04
	p-value	1.89E-02$^+$	**4.78E-02**$^+$	1.60E-09$^-$	1.42E-09$^-$	1.41E-09$^-$	-
F_{16}	Mean	3.00E+00	5.00E+00	3.00E+00	1.00E+00	2.00E+00	2.00E+00
	p-value	5.50E-03$^+$	**1.41E-09**$^+$	4.54E-07$^-$	1.04E-08$^-$	3.07E-04$^-$	-
F_{17}	Mean	5.50E+02	7.80E+02	8.40E+02	1.39E+03	1.98E+02	4.94E+02
	p-value	**2.80E-03**$^+$	**1.41E-09**$^+$	**1.80E-09**$^+$	**1.42E-09**$^+$	1.42E-09$^-$	-
F_{18}	Mean	1.20E+03	9.10E+02	2.53E+03	1.51E+03	3.09E+02	8.98E+02
	p-value	**1.99E-07**$^+$	**2.50E-03**$^+$	**1.42E-09**$^+$	**1.42E-09**$^+$	1.60E-09$^-$	-
F_{19}	Mean	4.90E+01	1.10E+01	8.20E+01	5.70E+02	1.80E+01	1.00E+01
	p-value	**1.41E-09**$^+$	5.35E-01$^=$	**1.41E-09**$^+$	**1.41E-09**$^+$	**5.21E-09**$^+$	-
F_{20}	Mean	5.00E+01	4.99E+01	4.99E+01	4.99E+01	5.00E+01	4.75E+01
	p-value	**1.04E-02**$^+$	**4.28E-02**$^+$	1.18E-01$^=$	3.97E-01$^=$	**1.04E-02**$^+$	-
F_{21}	Mean	3.30E+02	3.70E+02	4.20E+02	4.40E+02	4.10E+02	3.90E+02
	p-value	5.86E-01$^=$	3.17E-01$^=$	**5.17E-08**$^+$	**4.39E-09**$^+$	**5.12E-07**$^+$	-
F_{22}	Mean	6.97E+03	1.40E+03	5.05E+03	1.56E+04	5.48E+03	2.18E+03
	p-value	**1.42E-09**$^+$	8.86E-06$^-$	**1.42E-09**$^+$	**1.42E-09**$^+$	**1.42E-09**$^+$	-
F_{23}	Mean	2.38E+04	2.88E+04	2.57E+04	2.01E+04	1.10E+04	2.20E+04
	p-value	7.86E-01$^=$	1.04E-01$^=$	6.55E-01$^=$	2.98E-02$^-$	2.98E-02$^-$	-
F_{24}	Mean	5.50E+02	3.00E+02	6.10E+02	5.80E+02	3.70E+02	2.05E+02
	p-value	**1.42E-09**$^+$	**2.57E-09**$^+$	**1.42E-09**$^+$	**1.42E-09**$^+$	**1.80E-09**$^+$	-
F_{25}	Mean	6.30E+02	4.10E+02	6.00E+02	7.10E+02	4.60E+02	3.43E+02
	p-value	**1.42E-09**$^+$	**2.87E-08**$^+$	**1.42E-09**$^+$	**1.42E-09**$^+$	**1.42E-09**$^+$	-
F_{26}	Mean	6.20E+02	3.90E+02	6.80E+02	5.40E+02	4.40E+02	2.10E+02
	p-value	**1.42E-09**$^+$	**1.60E-09**$^+$	**1.42E-09**$^+$	**5.91E-05**$^+$	**1.42E-09**$^+$	-
F_{27}	Mean	3.56E+03	1.33E+03	4.45E+03	3.75E+03	1.99E+03	7.80E+02
	p-value	**1.42E-09**$^+$	**6.80E-07**$^+$	**1.42E-09**$^+$	**1.42E-09**$^+$	**2.87E-08**$^+$	-
F_{28}	Mean	4.72E+03	2.89E+03	1.76E+04	8.90E+03	3.38E+03	1.54E+03
	p-value	**1.99E-07**$^+$	**8.87E-05**$^+$	**1.41E-09**$^+$	**1.41E-09**$^+$	**4.67E-06**$^+$	-
w/l/t		25/1/2	18/4/6	24/2/2	23/3/2	20/7/1	-

indicate that the results of HPSO-GA are significantly better, significantly worse, or statistically similar to the results obtained by the corresponding peer algorithms, respectively. The best results of the average performance are highlighted by gray.

As shown by the results, HPSO-GA wins 16 times out of the 28 benchmarks with respect to the mean results. With a deeper insight, for the five unimodal functions F_1 to F_5, HPSO-GA wins for 3 times; for the basic multimodal functions F_6 to F_{20}, HPSO-GA wins for 8 times; for the composition functions F_{21} to F_{28}, HPSO-GA wins for 5 times. The statistic analysis results also indicate the competitive performance of HPSO-GA: it significantly outperforms the peer algorithms for 25, 18, 23, 24, and 20 times, respectively.

Therefore, one can see that the proposed algorithm is competitive in large-scale optimization (functions with dimensionality of 100). This can be explained by (i) the proposed CGCO is more suitable for convergence than PSO's position update strategy, since the information of the promising solutions can be directly fed into the updated particles; (ii) CGCO is able to enhance the diversity preservation ability for the proposed algorithm by diverse the exemplars; (iii) the proposed CGCO only updates half of the particles at each generation, which also results in a good diversity preservation ability; (iv) the proposed adaptive mutation operator can effectively further improve the diversity preservation ability for the proposed algorithm.

4.2. Curriculum Sequencing Optimization. In this section, a curriculum sequencing model is proposed. Afterwards, the proposed HPSO-GA and three algorithms are tested on the proposed model.

4.2.1. Problem Description. Traditional e-learning systems are neither reusable nor adaptive [3]. Furthermore, current e-learning systems commonly fix the learning paths for each student and are ineffective in providing adaptive learning schemes. A main reason is that the curriculum sequencing problem is a NP-hard optimization problem, resulting in difficulties to e-learning provider for properly scheduling the courses for students. Targeting against this issue, various kinds of curriculum sequencing models and optimization algorithms have been proposed [42, 43].

The curriculum sequencing problem can be commonly formulated by (L, DV, CT), where $L = l_1, l_2, \ldots, l_n$ represents the students with three properties, for instance, including the student's available time $l_i(t)$, student's ability $l_i(a)$, and student's objectives $l_i(O_k)$; DV is a mapping function which maps a student l_i to a finite ordered set of learning objectives (LOs) that can be assigned to l_i. CT is the constraints. The goal of the curriculum sequencing is that scheduling courses for students and satisfying all the constraints as much as possible.

In this study, we build the curriculum sequencing model by taking the following constraints into considerations.

(1) Various learning objectives for students
(2) International cooperation teaching
(3) Two-side satisfaction, e.g., both the requirements of teachers and students should be satisfied
(4) Suitable strength, e.g., students should be assigned with courses that not exceed their ability level

Following the above concerns, the proposed model is constructed as equations (5)–(9):

$$C_{\text{time},1} = \frac{\sum_i^{LN} \text{sign}\left(A_{\text{Time}} | \sum l_{i,\text{Time}}\right)}{LN}, \tag{5}$$

$$C_{\text{time},2} = \frac{\sum_i^{LN} \prod_k^{CN} \text{sign}\left(A_{\text{Time},k} | \sum l_{i,\text{Time}}\right)}{LN}, \tag{6}$$

$$C_{\text{stun}} = \frac{\sum_i^{CN} \text{sign}\left(A_{\text{course},i} | \sum \text{stu}_{\text{num}}\right)}{CN}, \tag{7}$$

$$C_{\text{diff}} = \frac{\sum_i^{LN} \prod_k^{CN} \text{sign}\left(A_{\text{course},k,\text{diff}} | l_{i,\text{diff}}\right)}{LN}, \tag{8}$$

$$C_O = \frac{\sum_i^{LN} \text{sign}\left(l_{i,o} | \sum_k^{CN} A_{\text{course},k,o}\right)}{LN}, \tag{9}$$

$$CS(\%) = C_{\text{time},1} + C_{\text{time},2} + C_{\text{stun}} + C_{\text{diff}} + C_O, \tag{10}$$

where $C_{\text{time},1}$ and $C_{\text{time},2}$ are the time constraints for students, which are used to examine whether the assigned total courses' time and each course fits the students' available time, C_{stun} is the maximum number of students that a course can fit, C_{diff} means whether the courses assigned to a student can fit the difficulty level that the student is in, C_O means whether the courses assigned to a student can meet his/her learning goals, CN and LN are the course number assigned to the corresponding student and the number of all the students, respectively, A_+Time and $A_{\text{time},k}$ are the total time of the courses and the time of the kth course, $A_{\text{course},i}$ is the maximum number of the students that the ith course can fit, stu_{num} is the corresponding number of students assigned to the ith course, $l_{i,o}$ is the learning objectives of the ith student while $A_{\text{course},k,o}$ is the objective involved in the kth course, $\text{sign}(\text{condition}_1 | \text{condition}_2)$ means whether the two conditions are fit to each other, if it is, it returns 1 or it returns 0; finally, the goal of the curriculum sequencing problem is to maximize the sum of the satisfaction degree (%) of the five constraints.

With the above designs, the proposed model can ensure that (i) the requirements of the students can be maximized and (ii) the time availability of the different students (e.g., the students live in different countries, resulting in different time availability) can be taken into considerations. Note that all the above constraints are built on 10 days period; such design is flexible to the student; they can dynamically change their learning goals and course difficulty level according to their improvements.

4.2.2. Simulation and Results. To test the proposed algorithm and model, comparison experiments are conducted with the following settings.

(1) Model Settings. First, to build the dataset, a real e-learning dataset from an information technology diploma program is considered, where 1000 students are randomly selected. The dataset is obtained from the corresponding student affairs' system. Each student is represented by an ID, a set of learning objectives, and an ability indicator and total/sectional available time as follows: $S = \{$ID: integer, objective: integer vector, ability: integer ranging between 1 and 5, total/sectional available time: integer, integer vector$\}$. Here, we randomly select 30% students as the foreign students with a time difference of 12 hours, while the maximum number of students that a course can fit is set to 30. Each course is represented by its ID, difficulty level, required time, and covered learning objectives as follows: Course = {ID: integer, difficulty level: integer ranging between 1 and 5, time required: integer, objective: integer vector}. Each solution of the optimization algorithms is coded by integer strategy and has a fixed dimensionality (the number of students multiplied by the maximum number of the courses that a student can take); the maximum number of the courses that a student can take during 10 days is set to 80. The time period adopted to conduct the experiments is 60 days. Furthermore, we only randomly select three learning objectives for each student in the comparison and test the scalability of the algorithms with respect to different number of the learning objectives.

(2) Compared Algorithms. Three algorithms are selected in the comparison: genetic algorithm, SwarmRW, and SwarmRW-rnd [3]. For the genetic algorithm, the crossover rate is set to 1 and the mutation rate is set to 0.001; for the two swarm-based algorithms, the suggested parameter settings in the corresponding references are adopted; for the proposed HPSO-GA, the crossover rate is set to 1 and the maximum mutation rate is set to 0.05. The swarm size for each algorithm is set to 200 due to the large number of decision variables; the maximum FEs is set to $3E + 06$. The Wilcoxon rank sum test is adopted to conduct the statistical test, where the significance level is set to 0.05.

(3) Results. Figure 3 shows the comparisons between HPSO-GA and the peer algorithms with respect to CS (%). As one can see, HPSO-GA outperforms GA, SwarmRW, and SwarmRW-rnd by more than 26 %, 11 %, and 7 %, respectively. The corresponding Wilcoxon rank sum test results show that the HPSO-GA significantly outperforms the compared algorithms: pvalue = $4.21E - 10$ between HPSO-GA and GA, pvalue = $3.55E - 04$ between HPSO-GA and SwarmRW, and pvalue = $1.26E - 08$ between HPSO-GA and SwarmRW-rnd.

Furthermore, the runtime of the algorithms is shown in Figure 4. One can see that SwarmRW is of the most computational simplicity, followed by the GA. HPSO-GA and SwarmRW-inc are comparable with respect to the runtime. However, in general, the runtime of HPSO-GA is acceptable in comparison to the GA and SwarmRW, and it brings improvements in the optimization results.

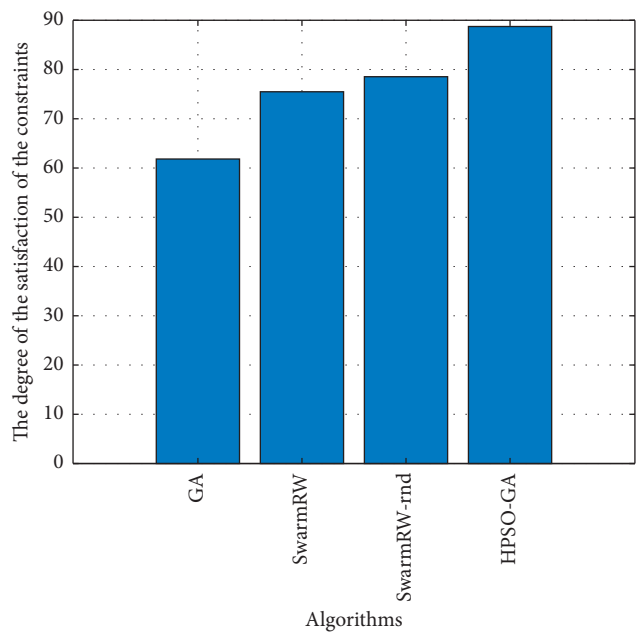


Figure 3: Comparison between HPSO-GA and the peer algorithms with respect to the CS (%).

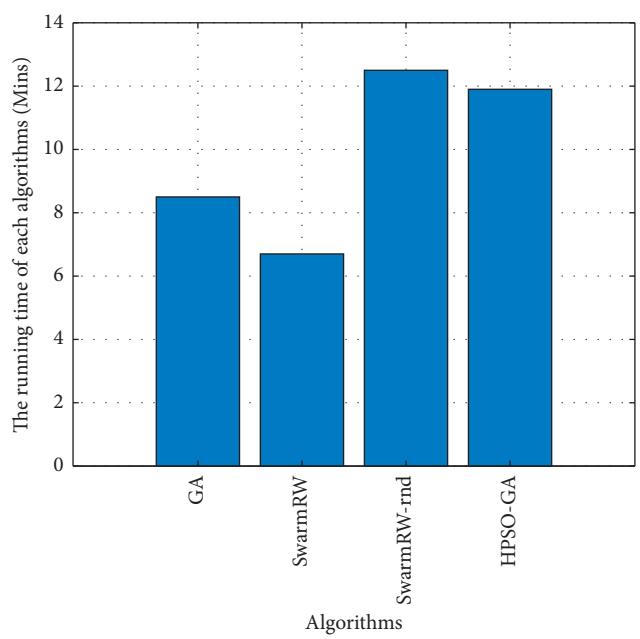


Figure 4: Comparison between HPSO-GA and the peer algorithms with respect to the runtime.

In addition, the satisfaction of individual constraint obtained by different algorithms is shown in Table 2. As shown by the results, HPSO-GA obtains the best results for 4 times out of the 5 constraints, followed by SwarmRW-rnd wins on one constraint. The results demonstrate that the proposed HPSO-GA is not only promising in the overall goal but also competitive in individual constraints.

Table 2: The comparisons on individual constraint satisfaction (%) of different algorithms.

Functions	Property	GA	SwarmRW	SwarmRW-rnd	HPSO-GA
$C_{time,1}$	Mean	58.81	78.92	79.87	90.14
	Std	3.87	4.15	3.66	3.98
$C_{time,2}$	Mean	63.76	74.78	78.09	87.88
	Std	3.12	2.67	3.22	3.03
C_{stun}	Mean	66.33	80.65	90.12	89.88
	Std	1.67	2.21	1.89	2.56
C_{diff}	Mean	58.65	70.45	71.34	86.86
	Std	2.76	2.34	3.11	3.78
C_O	Mean	61.59	72.62	73.28	88.91
	Std	1.11	2.84	3.96	3.48

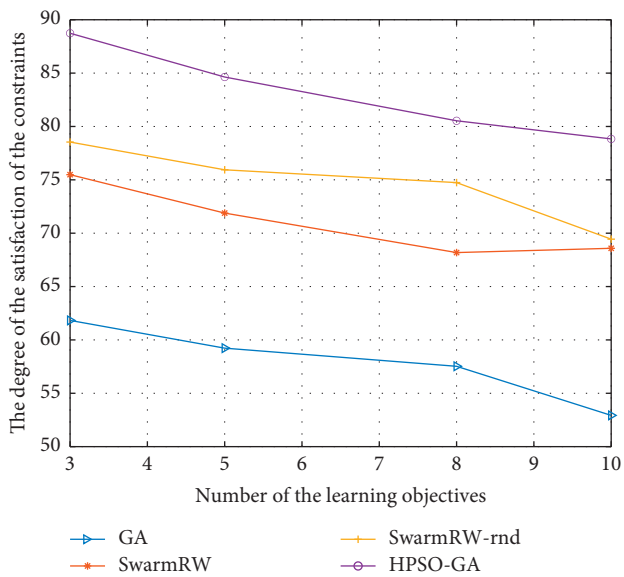


Figure 5: The scalability comparison between HPSO-GA and the peer algorithms with respect to the learning goals.

Finally, the scalability of different algorithms with respect to the variation of students' learning objective is tested; the results are shown in Figure 5. It can be observed that all the algorithms perform worse with the increasing of the learning objectives. This can be explained by that the difficulty of covering all the learning objectives of students becomes harder with the increasing of the number of the learning objectives. Therefore, it poses higher challenges to the searching ability of the algorithms. Nevertheless, the proposed HPSO-GA always outperforms other compared algorithms with respect to different number of learning objectives, which demonstrates the satisfied scalability of the proposed algorithm.

In summary, one can find that (i) the proposed curriculum sequencing model can be combined with real-world dataset and (ii) the proposed HPSO-GA is effective in solving the proposed model and provide proper curriculum paths for students.

5. Conclusions

Targeting against the modeling and optimization of the curriculum sequencing, this study first proposes a novel hybrid PSO algorithm by designing new particle update strategies, where a competitive-genetic crossover operator and an adaptive mutation operator are proposed for both convergence and diversity preservation. In the experiments, the numerical comparisons show that the proposed algorithm is competitive in comparison to several peer algorithms, indicating the effectiveness of the proposed strategies. On the contrary, several constraints are taken into considerations to model the curriculum sequencing problem, where the sectional available time expends the reliability of the proposed model for international e-learning systems.

Furthermore, the period-based model strategy is potentially beneficial to students to dynamically change their learning strategies, which is a future study point of this study.

Data Availability

The data are not available due to the nature of this research as participants of this study did not agree for their data to be shared publicly.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the Study on the Mechanism of Industrial-Education Co-Cultivation for Interdisciplinary Technical and Skilled Personnel in Chinese Intelligent Manufacturing Industry (Planning project for the 14th Five-Year Plan of National Education Sciences (BJA210093)).

References

[1] A. Y. Alqahtani and A. A. Rajkhan, "E-learning critical success factors during the covid-19 pandemic: a comprehensive analysis of e-learning managerial perspectives," *Education Sciences*, vol. 10, no. 9, p. 216, 2020.

[2] M. Ebner, S. Schön, C. Braun et al., "COVID-19 epidemic as E-learning boost? Chronological development and effects at an Austrian university against the background of the concept of "E-Learning readiness"," *Future Internet*, vol. 12, no. 6, p. 94, 2020.

[3] M. E. B. Menai, H. Alhunitah, and H. A. Salman, "Swarm intelligence to solve the curriculum sequencing problem," *Computer Applications in Engineering Education*, vol. 26, no. 5, pp. 1393–1404, 2018.

[4] L. D. Marcos, J. J. Martínez, and J. A. Gutiérrez, "Swarm intelligence in e-learning: a learning object sequencing agent based on competencies," in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, pp. 17–24, Atlanta, Georgia, July 2008.

[5] L. H. Wong and C. K. Looi, "A survey of optimized learning pathway planning and assessment paper generation with swarm intelligence," in *Intelligent Tutoring Systems in E-Learning Environments: Design, Implementation and Evaluation*IGI Global, Hershey, PA, USA, 2011.

[6] L. H. Wong and C. K. Looi, "Swarm intelligence: new techniques for adaptive systems to provide learning support," *Interactive Learning Environments*, vol. 20, no. 1, pp. 19–40, 2012.

[7] E. Kurilovas, I. Zilinskiene, and V. Dagiene, "Recommending suitable learning scenarios according to learners' preferences: an improved swarm based approach," *Computers in Human Behavior*, vol. 30, pp. 550–557, 2014.

[8] S. Al-Muhaideb and M. E. B. Menai, "Evolutionary computation approaches to the curriculum sequencing problem," *Natural Computing*, vol. 10, no. 2, pp. 891–920, 2011.

[9] L. de Marcos, R. Barchino, J. Martínez, and J. Gutiérrez, "A new method for domain independent curriculum sequencing: a case study in a web engineering master program," *International Journal of Engineering Education*, vol. 25, no. 4, p. 632, 2009.

[10] L. de Marcos, R. Barchino, J. J. Martínez, J. A. Gutiérrez, and J. R. Hilera, "Competency-based intelligent curriculum sequencing: comparing two evolutionary approaches,"vol. 3, pp. 339–342, in *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, vol. 3, IEEE, Sydney, Australia, December 2008.

[11] A. F. Martins, M. Machado, H. S. Bernardino, and J. F. de Souza, "A comparative analysis of metaheuristics applied to adaptive curriculum sequencing," *Soft Computing*, vol. 25, no. 16, pp. 11019–11034, 2021.

[12] D. Li, W. Guo, L. Wang, and M. Chen, "Particle swarm optimization-based solution updating strategy for biogeography-based optimization," in *Proceedings of the 2016 IEEE congress on Evolutionary Computation (CEC)*, pp. 455–459, IEEE, Vancouver, Canada, July 2016.

[13] D. Li, W. Guo, A. Lerch, Y. Li, L. Wang, and Q. Wu, "An adaptive particle swarm optimizer with decoupled exploration and exploitation for large scale optimization," *Swarm and Evolutionary Computation*, vol. 60, Article ID 100789, 2021.

[14] D. Li, W. Guo, L. Wang, and Q. Wu, "A modified apsodee for large scale optimization," in *Proceedings of the 2021 IEEE Congress on Evolutionary Computation (CEC)*, p. 1976, June 2021.

[15] Y. l. Zheng, L. h. Ma, L. y. Zhang, and J. x. Qian, "Empirical study of particle swarm optimizer with an increasing inertia weight,"vol. 1, pp. 221–226, in *Proceedings of the 2003 Congress on Evolutionary Computation, 2003. CEC'03*, vol. 1, pp. 221–226, IEEE, Canberra, Australia, December 2003.

[16] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[17] A. Chatterjee and P. Siarry, "Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization," *Computers & Operations Research*, vol. 33, no. 3, pp. 859–871, 2006.

[18] Z. H. Zhan, J. Jun Zhang, Y. Yun Li, and H. S. H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1362–1381, 2009.

[19] M. Taherkhani and R. Safabakhsh, "A novel stability-based adaptive inertia weight for particle swarm optimization," *Applied Soft Computing*, vol. 38, pp. 281–295, 2016.

[20] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

[21] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[22] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 191–204, 2014.

[23] Y. Chen, L. Li, J. Xiao, Y. Yang, J. Liang, and T. Li, "Particle swarm optimizer with crossover operation," *Engineering Applications of Artificial Intelligence*, vol. 70, pp. 159–169, 2018.

[24] R. L. Tang, Z. Wu, and Y. J. Fang, "Adaptive multi-context cooperatively coevolving particle swarm optimization for large-scale problems," *Soft Computing*, vol. 21, no. 16, pp. 4735–4754, 2017.

[25] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization,"vol. 3, pp. 1945–1950, in *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, vol. 3, pp. 1945–1950, IEEE, Washington, DC, USA, July 1999.

[26] Y. Shi and R. C. Eberhart, "Fuzzy adaptive particle swarm optimization,"vol. 1, pp. 101–106, in *Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546)*, vol. 1, pp. 101–106, IEEE, Seoul, South Korea, May 2001.

[27] A. P. Piotrowski, J. J. Napiorkowski, and A. E. Piotrowska, "Population size in particle swarm optimization," *Swarm and Evolutionary Computation*, vol. 58, Article ID 100718, 2020.

[28] D. Tian and Z. Shi, "Mpso: modified particle swarm optimization and its applications," *Swarm and Evolutionary Computation*, vol. 41, pp. 49–68, 2018.

[29] W.-N. Chen, J. Zhang, Y. Lin et al., "Particle swarm optimization with an aging leader and challengers," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 2, pp. 241–258, 2013.

[30] R. Cheng, C. Sun, and Y. Jin, "A Multi-Swarm Evolutionary Framework Based on a Feedback Mechanism," in *Proceedings of the 2013 IEEE Congress on Evolutionary Computation*, pp. 718–724, IEEE, Cancun, Mexico, June 2013.

[31] R. Cheng and Y. Jin, "A social learning particle swarm optimization algorithm for scalable optimization," *Information Sciences*, vol. 291, pp. 43–60, 2015.

[32] Q. Yang, W. N. Chen, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "A level-based learning swarm optimizer for large-scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 4, pp. 578–594, 2018.

[33] X. Zhang, H. Liu, and L. Tu, "A modified particle swarm optimization for multimodal multi-objective optimization," *Engineering Applications of Artificial Intelligence*, vol. 95, Article ID 103905, 2020.

[34] N. Zeng, Z. Wang, W. Liu, H. Zhang, K. Hone, and X. Liu, "A dynamic neighborhood-based switching particle swarm optimization algorithm," *IEEE Transactions on Cybernetics*, pp. 1–12, 2021.

[35] F. vandenBergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[36] F. V. D. Bergh, *An Analysis of Particle Swarm Optimizers*, Ph.D. thesis, University of Pretoria, Pretoria, South Africa, 2007.

[37] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 2, pp. 210–224, 2011.

[38] Q. Qin, S. Cheng, Q. Zhang, L. Li, and Y. Shi, "Particle swarm optimization with interswarm interactive learning strategy," *IEEE Transactions on Cybernetics*, vol. 46, no. 10, pp. 2238–2251, 2016.

[39] Y.-J. Gong, J.-J. Li, Y. Zhou et al., "Genetic learning particle swarm optimization," *IEEE Transactions on Cybernetics*, vol. 46, no. 10, pp. 2277–2290, 2016.

[40] K. Chen, F. Y. Zhou, and X. F. Yuan, "Hybrid particle swarm optimization with spiral-shaped mechanism for feature selection," *Expert Systems with Applications*, vol. 128, pp. 140–156, 2019.

[41] S. Mirjalili, "Genetic algorithm," *Studies in Computational Intelligence, in: Evolutionary Algorithms and Neural Networks*, Springer, New York, NY, USA, pp. 43–55, 2019.

[42] C. Chen, C. Liu, and M. Chang, "Personalized curriculum sequencing utilizing modified item response theory for web-based instruction," *Expert Systems with Applications*, vol. 30, no. 2, pp. 378–396, 2006.

[43] M. D. O. C. Machado, N. F. S. Bravo, A. F. Martins, H. S. Bernardino, E. Barrere, and J. F. d. Souza, "Metaheuristic-based adaptive curriculum sequencing approaches: a systematic review and mapping of the literature," *Artificial Intelligence Review*, vol. 54, no. 1, pp. 711–754, 2021.

Hindawi
Discrete Dynamics in Nature and Society
Volume 2022, Article ID 8948729, 12 pages
https://doi.org/10.1155/2022/8948729

Research Article

A Decomposition-Based Harmony Search Algorithm for Multimodal Multiobjective Optimization

Wei Xu, Weifeng Gao, and Qianlong Dang

School of Mathematics and Statistics, Xidian University, Xi'an 710126, China

Correspondence should be addressed to Weifeng Gao; gaoweifeng2004@126.com

Received 18 March 2022; Accepted 21 April 2022; Published 25 May 2022

Academic Editor: Xueyi Wang

Multimodal multiobjective optimization problem (MMOP) is a special kind of multiobjective optimization problem (MOP) with multimodal characteristics, where multiple different Pareto optimal sets (PSs) map to the same Pareto optimal front (PF). To handle MMOPs, a decomposition-based harmony search algorithm (called MOEA/D-HSA) is devised. In MOEA/D-HSA, multiple individuals who are assigned to the same weight vector form a subpopulation for finding multiple different PSs. Then, an environmental selection method based on greedy selection is designed to dynamically adjust the subpopulation scale for keeping the population diversity. Finally, the modified harmony search algorithm and elite learning strategy are utilized to balance the diversity and convergence of the population. Experimental results on the CEC 2019 test suite reveal that MOEA/D-HSA has superior performance than a few state-of-the-art algorithms.

1. Introduction

MOPs with multiple objectives that require to be optimized simultaneously are frequently occurring in real-world applications [1, 2]. In general, an MOP is defined as follows [3]:

$$\begin{aligned} &\min_{X\in\Omega} F(X) = \left(f_1(X), \ldots, f_m(X)\right)^T, \\ &\qquad x_i \in \left(x_i^l, x_i^u\right), \quad i = 1, 2, \ldots, n, \end{aligned} \tag{1}$$

where Ω refers to the search space and $X = (x_1, \ldots, x_n)^T \in \Omega$ denotes an n-dimensional decision variable; $f_j(X)$ indicates the j-th ($j = 1, 2, \ldots m$) objective, and m refers to the amount of objectives; x_i^u and x_i^l are the upper and lower bounds of x_i, respectively. In an MOP, a solution X is said to dominate another solution Y, if and only if $\forall j \in \{1, 2, \ldots, m\}$, $f_j(X) \le f_j(Y)$, and $\exists k \in \{1, 2, \ldots, m\}$, $f_k(X) < f_k(Y)$. If there exists no solution that dominates X, then X is called a Pareto optimal solution [4, 5]. The set of all Pareto optimal solutions is called PS [6], which is denoted as X^*. The image of X^* in the objective space is termed as PF [7].

A large number of MOPs with multimodal properties exist in practical applications [8], such as scheduling problem [9], epsilon-efficient solutions problem [10], rocket engine problem [11], and 0/1 knapsack problem [12]. This category of problems is termed MMOPs in [13]. In MMOPs, multiple different PSs map to the same PF. As depicted in Figure 1, an MMOP with two different Pareto regions (i.e., Region1 and Region2) in the decision space is presented. In Figure 1, all Pareto regions map to the same Pareto front (i.e., *PF*) in the objective space. For example, $A_1 \in$ Region1 and $A_2 \in$ Region2 correspond to the same point A and $A_1 \neq A_2$. All PSs should be discovered in handling MMOPs.

In the past few years, numerous multimodal multiobjective evolutionary algorithms (MMOEAs) have been developed to tackle MMOPs, which can be grouped into the following types:

(i) Pareto-based MMOEAs: DN-NSGA-II [13], Niching-CMA [14], and Omni-optimizer [15] are the three typical algorithms in this class. DN-NSGA-II proposed a niching method to deal with MMOPs, in which the crowding distance is adopted to find multiple PSs. Unlike DN-NSGA-II, an omni-optimizer calculated the crowding distance in both the decision and objective spaces to locate the

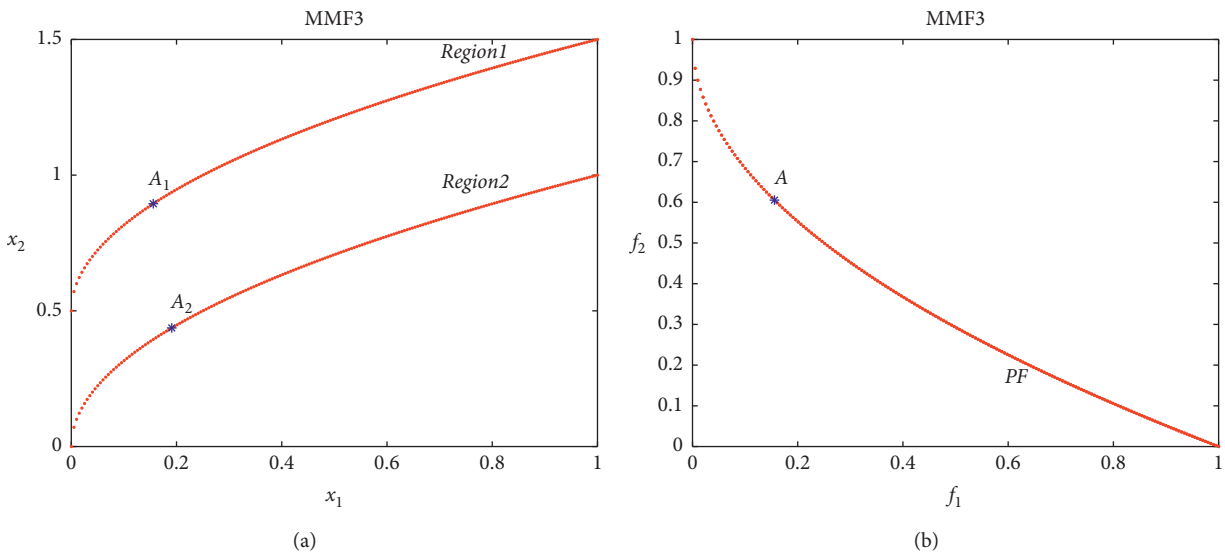


Figure 1: Example of MMF3 with two PSs to illustrate MMOP.

equivalent PSs. In Niching-CMA, a niche function is constructed to replace the crowding distance in the omni-optimizer by summing the distances of the individuals in the two spaces. Experimental results indicated that Niching-CMA can improve the diversity of decision space. Subsequently, a method similar to the omni-optimizer is proposed to deal with MMOPs (named MO_Ring_-PSO_SCD) [16], where the ring topology and diversity measure are utilized to keeping the population diversity. In addition, Liu et al. [17] proposed a penalty density strategy (called CPDEA) for solving MMOPs. In CPDEA, the individual density is calculated by the transformed Euclidean distance and used as the selection criteria to search for the equivalent PSs. Then, they proposed an MMOEA based on archives and recombination strategy to find multiple PSs (called TriMOEA-TA&R) [18]. Fan et al. [19] devised a zoning search approach (ZS) to tackle MMOPs, where the whole space is classified into multiple subspaces and existing MMOEAs are executed in each subspace. The results revealed that ZS helps to increase the property of MMOEAs in solving MMOPs. Inspired by this, Fan et al. proposed a zoning search approach with adaptive resource allocating [20] for balanced and imbalanced MMOPs. Lin et al. [21] presented a clustering-based MMOEA that is able to efficiently locate local PSs. Zhang et al. [22] devised a knee-based approach to deal with MMOPs, where a multicriteria decision strategy is introduced to search for equivalent PS regions. Afterwards, the repulsive force between isotropic magnetic individuals is introduced into MMOPs to locate different PSs by Zhang et al. [23, 24]. In addition, inspired by online learning [25, 26]. Li et al. [27] devised a differential evolution based on reinforcement learning to handle MMOPs. However, these algorithms generally adopt Pareto-based method to search for equivalent PSs, so they cannot deal with large-scale MMOPs.

(ii) Decomposition-based MMOEAs: like MOEA/D [28], an MMOP is decomposed into a few simple subproblems through a group of evenly distributed vectors in decomposition-based MMOEAs. In [29], the diversity maintenance strategy combining penalty boundary intersection technique [28] and two distance metrics is introduced into the decision space for locating equivalent PSs. Later, a similar concept is suggested to handle the large-scale optimization problems in [30]. Tanabe et al. [31] designed a modified MOEA/D for MMOPs, which employs the addition and deletion operators to search for equivalent PSs. In [32], this concept was expanded as a framework to increase the diversity of MMOEAs in the decision space. In [33], an evolutionary algorithm based on the graph Laplacian is presented to tackle MMOPs, which employs the decomposition technique in two spaces to obtain equivalent Pareto solutions. Although the above algorithms can obtain multiple different PSs, they are unable to balance the population diversity in the decision and objective spaces well.

(iii) Indicator-based MMOEAs: the indicator-based approaches utilize the chosen indicators to guide population converge to PF. Li et al. [34] proposed a weighted indicator-based MMOEA (termed MMEA-WI), which devised an indicator to keep the diversity in the decision space and introduced a convergence archive for approximating the true PF. In [35], a Niching indicator is designed to solve MMOPs, where individuals' fitness values are calculated to keep the diversity of PSs. In summary, the

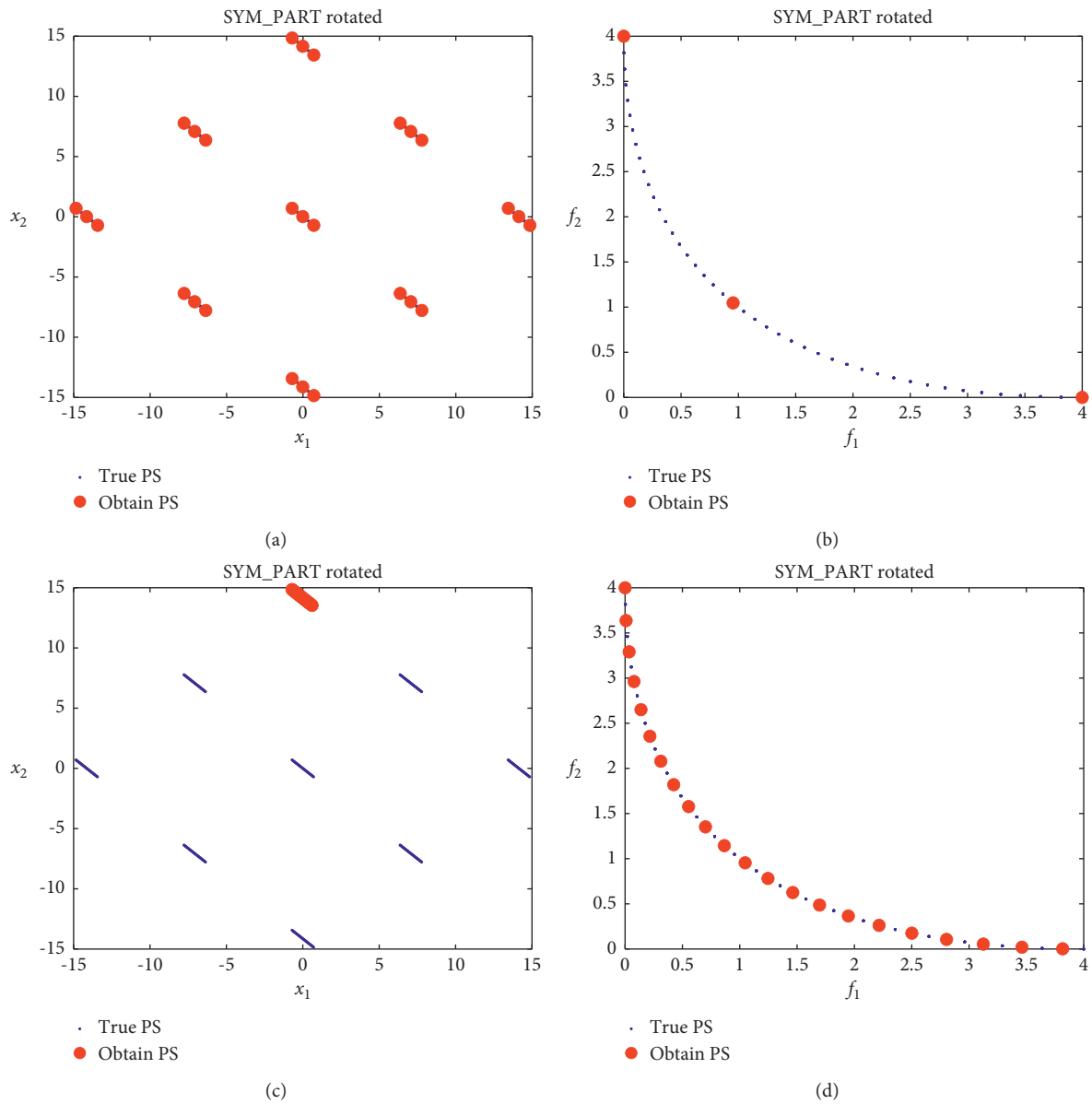


Figure 2: Example of SYM-PART rotated with nine PSs to explain the importance of diversity.

indicator-based MMOEAs usually adopt fitness indicator to keep the diversity in the decision space, but the calculation of indicator wastes a large amount of computing resources, especially for large-scale MMOPs:

To sum up, although the decomposition-based MMOEAs can find multiple equivalent PSs in dealing with MMOPs, they are unable to well balance the diversity in the decision and objective spaces. In other words, they increase the diversity in the decision space at the cost of the performance in the objective space. To deal with this issue, a decomposition-based harmony search algorithm (named MOEA/D-HSA) is presented. In MOEA/D-HSA, multiple individuals who are assigned to the same weight vector form a subpopulation after an MMOP is decomposed. Then, an environmental selection method is used for guiding the individuals in the subpopulation to find the equivalent PSs. Finally, the modified harmony search algorithm is utilized to generate the offspring, which helps to balance the relationship of diversity and convergence. At the same time, an elite learning strategy is employed to prevent premature convergence. The test results on CEC 2019 validate the property of MOEA/D-HSA.

2. Materials and Methods

2.1. Motivation. The diversity of decision and objective spaces is crucial in solving MMOPs. In the following, we take SYM-PART rotated with nine distinct PSs as an example to explain the importance of the diversity in these two spaces. As depicted in Figures 2(a) and 2(b), all PSs are located in the decision space, but only three points are found on PF. The

reason is that the algorithm pays more attention to the diversity in the decision space. For Figures 2(c) and 2(d), although the obtained PF has well-diversity and well-convergence, only one PS is located. When an algorithm is only concerned about the diversity in the objective space, the situation in Figures 2(c) and 2(d) may appear. Therefore, the following issues need to be considered for preventing situations in Figure 2.

(1) How to preserve the diversity in the objective space? This problem has been widely studied in multi-objective evolutionary algorithms (MOEAs). The classic methods include the weighted Tchebycheff approach [28], nondominated sorting method [36], and the fitness indicator technology [37, 38]. The weighted Tchebycheff approach is selected to deal with this problem. The reason is that the whole objective space is classified into multiple subspaces by a group of evenly distributed weight vectors in the weighted Tchebycheff approach. Thus, it can keep the diversity in this space. Besides, for balancing the exploration and exploitation of the algorithm, a modified harmony search algorithm is adopted to generate the offspring, where the algorithm parameters such as harmony memory size (HMS), harmony memory consideration rate (HMCR), and pitch adjusting rate (PAR) are dynamically adjusted.

(2) How to preserve the diversity in the decision space? For handling this problem, the framework of subpopulation is adopted to search for different PSs, i.e., each weight vector corresponds to multiple individuals instead of one. As shown in Figure 3, an MMOP is decomposed into a few simple subproblems by a group of evenly distributed weight vectors W_j $(j = 1, 2, \ldots, i)$, and each weight vector is given to three distinct individuals to shape a subpopulation (i.e., the red, yellow, and blue dots indicate distinct individuals who are assigned to the same weight vector). In this way, the individuals corresponding to the same weight vector search toward the equivalent PSs. Furthermore, preserving the diversity of the subpopulation is vital in MMOEAs. Therefore, an environmental selection method is designed to separate individuals belonging to the same PS in the subpopulation.

2.2. The Proposed Algorithm. A decomposition-based harmony search algorithm for MMOPs, termed MOEA/D-HSA, is presented. In MOEA/D-HSA, first, initialize population P, weight vector W, and subpopulation T_i $(i = 1, 2, \ldots, NP)$ (lines 1–6). Then, multiple individuals that are assigned to the same weight vector form a subpopulation (lines 7–9). Next, an offspring μ is generated by a modified harmony search algorithm and elite learning strategy (line 13). Lastly, the subpopulation is dynamically adjusted by an environmental selection method, which helps to preserve the diversity of the decision space (line 14). Algorithm 1 depicts the framework of MOEA/D-HSA.

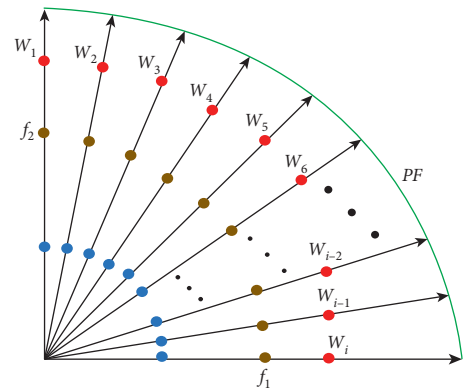


Figure 3: Illustration of the subpopulation framework.

2.3. Reproduction. For balancing the diversity and convergence of the population, a modified harmony search algorithm [39] shown in Algorithm 2 is used to generate the offspring. In Algorithm 2, first, initialize the algorithm parameters (i.e., HMS, HMCR, PAR, and BW). Then, parameters HMS, HMCR, and PAR are updated as follows [40, 41]:

$$HMS = HMS_{\max} - \left\lceil \frac{FES}{\mathrm{Max}FES} \right\rceil \times (HMS_{\max} - HMS_{\min}), \tag{2}$$

$$HMCR = HMCR_{\max} - \frac{(HMCR_{\max} - HMCR_{\min})}{\mathrm{Max}FES} \cdot FES, \tag{3}$$

$$PAR = PAR_{\min} + \frac{(PAR_{\max} - PAR_{\min})}{\mathrm{Max}FES} \cdot FES, \tag{4}$$

where MaxFES is the maximum number of evaluation functions, FES is the current number of evaluation functions, $\mathrm{HMS} \in [\mathrm{HMS}_{\min}, \mathrm{HMS}_{\max}]$, $\mathrm{HMCR} \in [\mathrm{HMCR}_{\min}, \mathrm{HMCR}_{\max}]$ and $\mathrm{PAR} \in [\mathrm{PAR}_{\min}, \mathrm{PAR}_{\max}]$. With the help of adaptive parameters, the proposed algorithm has more extensive exploration ability in the early phase of evolution and nice exploitation ability in the late phase of evolution. It means that MOEA/D-HSA can well balance the relationship of diversity and convergence. Next, the harmony memory (HM) containing HMS individuals is constructed. Finally, the offspring μ is generated.

2.4. The Elite Learning Strategy. For avoiding premature convergence, the elite learning strategy is employed to generate the offspring μ, which can increase the global search ability and help the algorithm to jump out of local optimal location.

$$x_{\mathrm{new},j} = x_{\mathrm{best},j} + \mathrm{Gauss}\left(0, pr^2\right) \times \left(x_j^u - x_j^l\right) \cdot \mathrm{if} \cdot \mathrm{rand} < pr, \tag{5}$$

where $\mathrm{Gauss}(0, pr^2)$ is a Gaussian distribution with mean 0 and standard deviation pr, $x_{\mathrm{best},j}$ is the j-th dimension variable of the best individual in HM, and x_j^u and x_j^l are the

Input: population size *NP,* decision variable dimension *n*, algorithm parameters: *HMS*, *HMCR*, *PAR*, bandwidth (*BW*), *pr, t*, the maximum number of evaluation functions *MaxFES*,
Output: the set *TP.*

```
Initialize population P and weight vector W;
Calculate the fitness Fit and ideal point Z*;
Set FES = NP;
for i = 1:NP do
    Set T_i = [];
end for
for i = 1:NP do
    T_j = T_j ∪ X_i, where j = min{g^tch(X_i|W_1, Z*), ..., g^tch(X_i|W_NP, Z*)};
end for
while FES < MaxFES do
    Set T = T_1 ∪ ··· ∪ T_NP;
    for i = 1:|T| do
        Generate offspring μ by Algorithm 2 and elite learning strategy;
        Update ideal point Z* and subpopulation T_j by Algorithm 3, where j = min{g^tch(μ|W_1, Z*), ..., g^tch(μ|W_NP, Z*)};
        FES = FES + 1;
end for
end while
TP ← all Pareto optimal solutions in T_1 ∪ ··· ∪ T_NP;
```

Algorithm 1: MOEA/D-HSA.

Input: archive *EXA*, individual X_i, algorithm parameters: HMS, HMCR, PAR, BW, the maximum number of evaluation functions Max*FES,* the current number of evaluation functions FES, decision variable dimension *n.*
Output: an offspring μ.

```
Update the parameters HMS, HMCR, and PAR using equations (2)–(4)
Select the nearest HMS individuals to X_i in EXA and put them into HM
for j = 1:n do
    if rand ≤ HMCR, then
        x_new,j ∈ {x_1,j, x_2,j, ..., x_HMS,j}
        if rand ≤ PAR, then
            x_new,j = x_new,j + rand · BW
        end if
    else
        x_new,j = x_j^l + rand · (x_j^u − x_j^l);
    end if
end for
μ = (x_new,1, ..., x_new,n)^T;
```

Algorithm 2: Reproduction.

upper and lower bounds of x_j, rand $\in (0,1)$. In the early phase of the algorithm, the larger pr value makes the offspring have a greater mutation range that is helpful for global search. In the late phase of the algorithm, small mutation range is helpful for local search. Therefore, pr is updated as follows:

$$pr = pr_{\max} - \frac{(pr_{\max} - pr_{\min})}{\text{Max}FES} FES, \tag{6}$$

where Max*FES* is the maximum number of evaluation functions, FES is the current number of evaluation functions, $pr_{\max}$ and $pr_{\min}$ are the upper and lower bounds of pr, and pr decreases as the iteration increases.

2.5. Environmental Selection. For MOEA/D-HSA, it is vital to keep the diversity of subpopulations in the decision space. To reach this goal, the environmental selection method presented in Algorithm 3 is employed to separate individuals belonging to the same PS in the subpopulation. In Algorithm 3, when the subpopulation scale exceeds t, the individual in the subpopulation that is nearest to the offspring μ is compared with μ by the weighted Tchebycheff approach and the better individuals are saved. Otherwise, μ into the next generation. In the following, we take the subpopulation T_j as an example to explain the working mechanism of environmental selection. As depicted in Figure 4, an MMOP has three

Input: offspring μ, Subpopulation T_j, algorithm parameter t.
Output: subpopulation T_j .
if $|T_j| + 1 > t$, then
  Select the nearest individual X to μ in T_j
  if $g^{tch}\{\mu|W_j, Z^*\} < g^{tch}\{X|W_j, Z^*\}$, then
    $T_j = T_j \backslash X$ and $T_j = T_j \cup \mu$
  end if
else
  $T_j = T_j \cup \mu$
end if

Algorithm 3: Environmental Selection

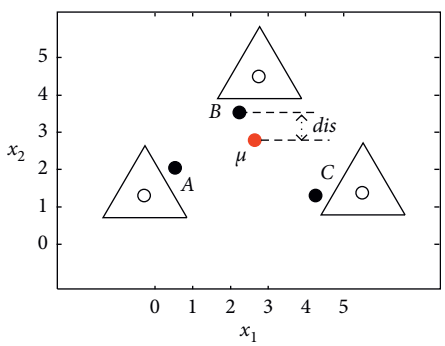


Figure 4: Example of subpopulation T_j to illustrate the working mechanism of environmental selection.

equivalent PSs, and all individuals within the triangle are Pareto optimal solutions. In Figure 4, the subpopulation T_j consists of four different individuals (i.e., A, B, C, and offspring μ). When the subpopulation scale is larger than t, the individual in the subpopulation T_j that is nearest to the offspring μ is compared with μ (here, B is the closest individual to μ), and the preferable one is preserved. Otherwise, μ into the subpopulation T_j. By this way, MOEA/D-HSA can find multiple different PSs.

2.6. Complexity Analysis. MOEA/D-HSA consists of the following components: initialization subpopulation, reproduction operation, and the environmental selection. The complexity of initialization subpopulation and environmental selection is $O(NP^2)$. The complexity of the reproduction operation is $O(n \times NP)$, where n is the decision variable dimension. In summary, considering $n < NP$, the complexity of MOEA/D-HSA is $O(NP^2)$.

3. Results and Discussion

3.1. Performance Metric and Benchmark Test Functions. In our experiment, 22 MMOPs from CEC 2019 competition [42] is utilized to examine the property of MMOEAs. In addition, inverted generational distance in the objective space (IGDF) [43, 44] and the Pareto set proximity (PSP) [16] are utilized to assess the property of algorithms. The lager the PSP is, the better the property of algorithm is. IGDF is just the opposite. They are defined as follows:

(1) IGDF

$$IGDF = \sum_{Y \epsilon PF} \frac{\min_{X \epsilon pf} ED(X, Y)}{|PF|}, \tag{7}$$

where $\mathrm{ED}(X, Y)$ denotes the Euclidean distance between X and Y, PF and pf refer to the true PF and the obtained PF, respectively. IGDF can assess the diversity and convergence of the obtained solutions in the objective space.

(2) PSP

$$PSP = \frac{\mathrm{CR}}{IGDX},$$
$$IGDX = \sum_{Y \epsilon PS} \frac{\min_{X \epsilon ps} ED(X, Y)}{|PS|}, \tag{8}$$

where IGDX is inverted generational distance in the decision space, PS and ps represent the true PS and the obtained PS, respectively, CR denotes the cover ratio between ps and PS and the details refer to [16], $\mathrm{ED}(X, Y)$ is the Euclidean distance between X and Y. PSP can reflect the convergence and diversity of the population in the decision space.

3.2. Competitive Algorithms and Parameter Setup. To test the efficiency of the algorithm, MOEA/D-HAS is compared with five up-to-date MMOEAs: DN-NSGA-II [13], TriMOEA-TA&R [18], LORD [33], MOEA/D [28], and MO_Ring_-PSO_SCD [16]. For a fair comparison, the population size is set to 800, all algorithms are independently implemented 25 times, and the maximum function evaluations are set to 80000 [16]. In addition, the parameter setting of all competitors is consistent with their original references. For MOEA/D-HSA, according to [40, 41, 45], the algorithm parameters are $\mathrm{HMCR}_{\max} = 0.9$, $\mathrm{HMCR}_{\min} = 0.2$, $\mathrm{PAR}_{\max} = 1$, $\mathrm{PAR}_{\min} = 0.6$, $\mathrm{HMS}_{\max} = 5$, $\mathrm{HMS}_{\min} = 2$, $\mathrm{BW} = 0.1$, $\mathrm{pr}_{\max} = 0.2$, $\mathrm{pr}_{\min} = 0.05$, and $t = 5$.

In further, the Wilcoxon's rank-sum test [46] and Friedman's test [47] are utilized to draw statistically reliable conclusions with significance level $\alpha = 0.05$. The symbols "+," "–," and "=" denote that MOEA/D-HSA is distinctly

Table 1: Experimental results of six algorithms for PSP.

	MMOEA/D-HSA	MO_Ring_PSO_SCD	TriMOEA-TA&R	DN-NSGA-II	LORD	MOEA/D
MMF1	**86.39**	65.76(+)	29.72(+)	47.67(+)	70.04(+)	8.62(+)
MMF1_e	**6.21**	5.43(+)	1.31(+)	0.81(+)	2.39(+)	0.75(+)
MMF1_z	**118.21**	90.04(+)	29.92(+)	48.32(+)	57.44(+)	8.36(+)
MMF2	**155.33**	105.86(+)	73.04(+)	68.85(+)	145.32(+)	4.31(+)
MMF3	**182.72**	138.59(+)	31.29(+)	84.24(+)	169.81(+)	5.74(+)
MMF4	**176.86**	112.72(+)	85.43(+)	38.46(+)	136.69(+)	2.19(+)
MMF5	**48.96**	33.71(+)	17.71(+)	14.83(+)	30.91(+)	3.86(+)
MMF6	**54.89**	35.39(+)	21.45(+)	18.46(+)	33.20(+)	4.76(+)
MMF7	**150.19**	108.12(+)	60.71(+)	96.55(+)	129.81(+)	8.27(+)
MMF8	**60.11**	46.20(+)	9.95(+)	17.01(+)	13.94(+)	0.11(+)
MMF9	567.62	331.47(+)	344.58(+)	36.57(+)	**648.92**(−)	0.96(+)
MMF10	0.71	**6.16**(−)	0.21(+)	0.13(+)	0.11(+)	0.07(+)
MMF11	0.59	**3.07**(−)	0.63(=)	0.46(+)	0.76(−)	0.11(+)
MMF12	0.52	**4.10**(−)	0.16(+)	0.32(+)	0.54(=)	0.14(+)
MMF13	1.92	**2.31**(−)	1.65(+)	1.77(+)	1.72(+)	0.97(+)
MMF14	**28.32**	26.95(+)	27.41(+)	14.99(+)	20.59(+)	7.75(+)
MMF14_a	**23.31**	23.01(=)	21.27(+)	11.70(+)	15.43(+)	9.39(+)
MMF15	4.65	**6.07**(−)	3.14(+)	3.37(+)	0.94(+)	0.68(+)
MMF15_a	4.81	**6.09**(−)	3.64(+)	4.10(+)	3.64(+)	3.63(+)
SYM_PART simple	21.61	20.67(+)	**76.69**(−)	0.69(+)	43.32(−)	0.01(+)
SYM_PART rotated	**19.19**	18.23(+)	13.66(+)	2.95(+)	17.97(+)	0.01(+)
Omni-test	11.47	7.16(+)	14.48(−)	1.09(+)	**26.17**(−)	0.03(+)
Sum up	+/ = /−	15/1/6	19/1/2	22/0/0	17/1/4	22/0/0

Table 2: Experimental results of six algorithms for *IGDF*.

	MMOEA/D-HSA	MO_Ring_PSO_SCD	TriMOEA-TA&R	DN-NSGA-II	LORD	MOEA/D
MMF1	7.442 $e-04$	9.938 $e-04$(+)	2.368 $e-03$(+)	8.638 $e-04$(+)	6.226 $e-04$(−)	6.562 $e-04$(−)
MMF1_e	1.882 $e-03$	2.948 $e-03$(+)	2.467 $e-03$(+)	9.669 $e-04$(−)	6.817 $e-04$(−)	2.068 $e-03$(+)
MMF1_z	6.461 $e-04$	9.501 $e-04$(+)	2.362 $e-03$(+)	7.424 $e-04$(+)	6.494 $e-04$(+)	5.188 $e-04$(−)
MMF2	3.449 $e-03$	5.188 $e-03$(+)	4.366 $e-03$(+)	**1.165 $e-03$**(−)	1.758 $e-03$(−)	6.826 $e-04$(−)
MMF3	2.917 $e-03$	3.805 $e-03$(+)	4.037 $e-03$ (+)	7.288 $e-04$(−)	1.576 $e-03$(−)	6.729 $e-04$(−)
MMF4	**4.578 $e-04$**	8.801 $e-04$ (+)	3.107 $e-02$(+)	7.881 $e-04$(+)	5.319 $e-04$(+)	4.781 $e-04$(+)
MMF5	**6.162 $e-04$**	9.423 $e-04$(+)	4.021 $e-03$ (+)	8.161 $e-04$(+)	6.203 $e-04$(+)	6.289 $e-04$(+)
MMF6	**5.757 $e-04$**	9.071 $e-04$(+)	2.252 $e-03$ (+)	8.218 $e-04$(+)	5.811 $e-04$(+)	6.296 $e-04$(+)
MMF7	**5.738 $e-04$**	9.158 $e-04$ (+)	3.131 $e-03$(+)	1.017e-03(+)	5.778e-04(+)	5.380 $e-04$(+)
MMF8	8.018 $e-04$	1.311 $e-03$ (+)	2.467 $e-03$(+)	8.668 $e-04$(+)	7.408 $e-04$(−)	5.290 $e-04$(−)
MMF9	8.235 $e-03$	3.658 $e-03$ (−)	6.178 $e-02$(+)	3.444 $e-03$(−)	2.489 $e-03$(−)	9.047 $e-03$(+)
MMF10	**1.901 $e-01$**	1.909 $e-01$(+)	2.263 $e-01$(+)	1.917 $e-01$(+)	1.914 $e-01$(+)	1.943 $e-01$(+)
MMF11	9.668 $e-02$	**7.584 $e-02$**(−)	1.574e-01(+)	9.036 $e-02$(−)	8.717 $e-02$(−)	9.768 $e-02$(+)
MMF12	8.332 $e-02$	**6.189 $e-02$**(−)	8.460 $e-02$(+)	8.398 $e-02$(+)	8.352 $e-02$(+)	8.379 $e-02$(+)
MMF13	1.503 $e-01$	**8.269 $e-02$**(−)	2.420 $e-01$(+)	1.458 $e-01$(=)	1.593 $e-01$(+)	1.520 $e-01$(+)
MMF14	**5.517 $e-02$**	5.867 $e-02$ (+)	8.526 $e-02$(+)	6.745 $e-02$(+)	5.598 $e-02$(+)	6.870 $e-2$(+)
MMF14_a	5.594 $e-02$	**5.385 $e-02$** (=)	7.541 $e-02$(+)	7.882 $e-02$(+)	5.410 $e-02$(=)	6.936 $e-02$(+)
MMF15	1.779 $e-01$	**1.530 $e-01$**(−)	2.058 $e-01$(+)	1.801 $e-01$(+)	1.844 $e-01$(+)	1.941 $e-01$(+)
MMF15_a	1.791 $e-01$	**1.553 $e-01$**(−)	1.954 $e-01$(+)	1.907 $e-01$(+)	1.840 $e-01$(+)	1.928 $e-01$(+)
SYM_PART simple	1.048 $e-02$	9.703 $e-03$ (−)	4.051 $e-02$(+)	**3.074 $e-03$**(−)	5.698 $e-03$(−)	8.199 $e-03$(−)
SYM_PART rotated	1.032 $e-02$	1.179 $e-02$ (+)	1.407 $e-02$(+)	**3.837 $e-03$**(−)	6.151 $e-03$(−)	8.199 $e-03$(−)
Omni-test	1.264 $e-02$	1.796 $e-02$ (+)	8.205 $e-03$(−)	**2.881 $e-03$**(−)	4.022 $e-03$(−)	4.208 $e-03$(−)
Sum up	+/ = /−	14/1/7	21/0/1	13/1/8	11/2/10	14/0/8

better, worse, and not much different from the other competitor, respectively.

3.3. Comparison with Other Algorithms. The experimental results about *PSP* are shown in Table 1. As indicated in Table 1, MOAE/D-HSA achieves the best PSP on 13 test functions and defeats DN-NSGA-II and MOEA/D on all test functions. MO_Ring_PSO_SCD gets the maximum *PSP* on six test functions out of 22. LORD acquires the best performance on two test functions. TriMOEA-TA&R gets the maximum PSP on one test function. Although MOAE/D-HSA is not the best on some test functions, the difference is appreciably small and acceptable. Therefore, the

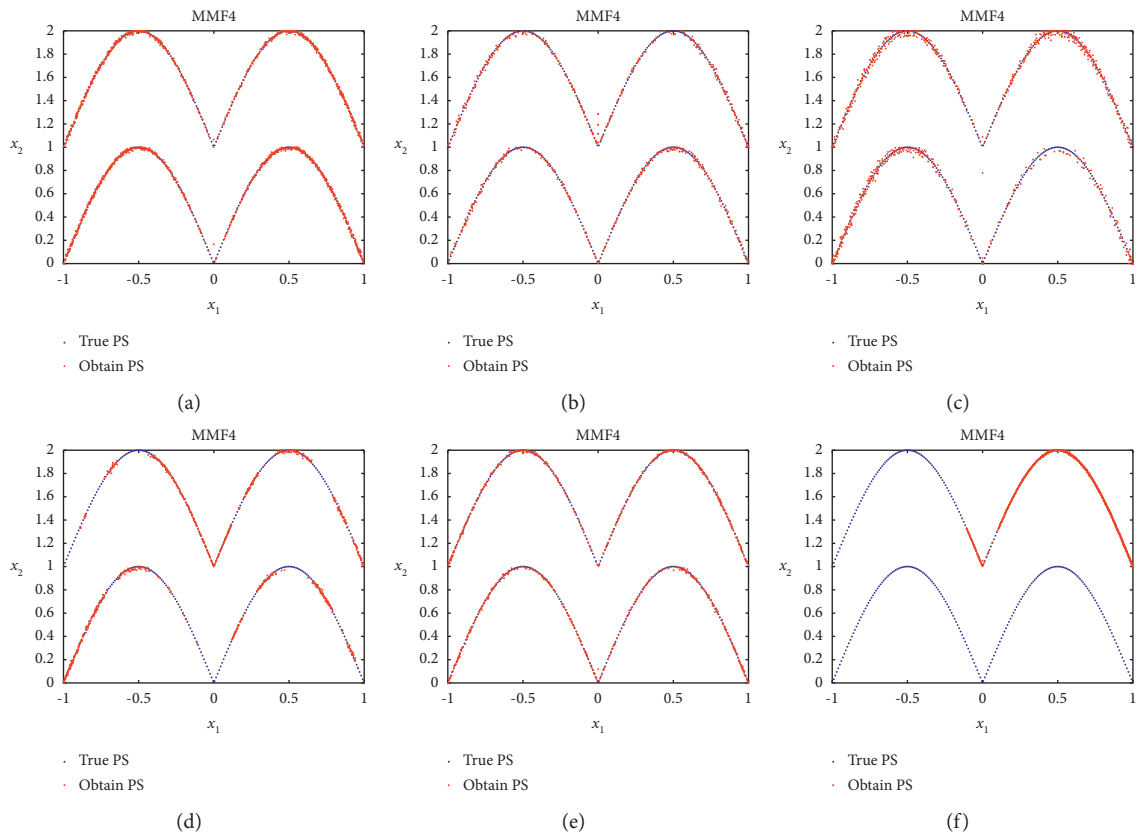


Figure 5: The population distributions of the true PSs and the final solutions obtained by different MMOEAs for MMF4 with four equivalent PSs, (a) MOEA/D-HSA, (b) MO_Ring_PSO_SCD, (c) TriMOEA-TA&R, (d) DN-NSGA-II, (e) LORD, and (f) MOEA/D.

experimental results suggest that MOAE/D-HSA can find multiple PSs with well-distribution and well-convergence.

The experimental results in IGDF are listed in Table 2. In Table 2, MOEA/D-HSA gets the smallest IGDF on six test functions and defeats TriMOEA-TA&R on most test functions. Moreover, MO_Ring_PSO_SCD is the best on six test functions out of 22. DN-NSGA-II acquires the best IGDF on four test functions. MOEA/D and LORD get the minimum *IGDF* on three test functions. In conclusion, it can be concluded that MOEA/D-HSA is not only outperforms the competitors on most test functions in IGDF but also can well approximate the true PF.

3.4. Visual Comparison. The final population distributions of all algorithms are shown in Figure 5 for MMF4 and Figure 6 for SYM_PART rotated. As illustrated in Figure 5, each algorithm except MOEA/D is able to locate all PS regions, but TriMOEA-TA&R has poor convergence in the decision space, and DN-NSGA-II achieves the least number of Pareto optimal solutions on each PS region. In addition, compared with other algorithms, not only MOEA/D-HSA can locate more Pareto optimal solutions on each PS but also the obtained PSs have well-diversity and well-convergence.

Besides, for SYM_PART rotated with nine equivalent PSs, as depicted in Figure 6 that all algorithms have good convergence in the decision space, but MOEA/D and DN-NSGA-II cannot locate all PSs. Furthermore, MO_Ring_PSO_SCD and MOEA/D-HSA have similar properties on SYM_PART rotated. And, compared with the remaining competitors, MOEA/D-HSA not only can search more Pareto optimal solutions on each PS but also show good convergence. In summary, based on visual comparison, MOEA/D-SS can find multiple PSs with good distribution and convergence.

To further verify the effectiveness of each component, two MOEA/D-HSA variants are test. SS-variant-1 indicates MOEA/D-HSA without the elite learning strategy. SS-variant-2 represents MOEA/D-HSA deletes an individual with the largest Tchebycheff value in the subpopulation instead of using the environmental selection operation. Table 3 records the average rankings of *PSP* obtained by MOEA/D-HSA and MOEA/D-HSA variants on all test functions. As depicted in Table 3, MOEA/D-HSA owns the best ranking. Therefore, it can be concluded that each component of the proposed algorithm can collaborate with the other components to enhance its overall performance.

3.5. Parameter Analysis. All parameters except t are adaptively adjusted, so only the subpopulation size t is discussed in this section. MOEA/D-HSA with $t = \{2, 3, 4, 5, 6\}$ is tested on all problems, and Table 4 records their rankings of the Friedman test. In Table 4, it can be seen that the property of MOEA/D-HSA is the best when t is set to 5. Therefore, the subpopulation size t is set to 5 in the proposed algorithm.

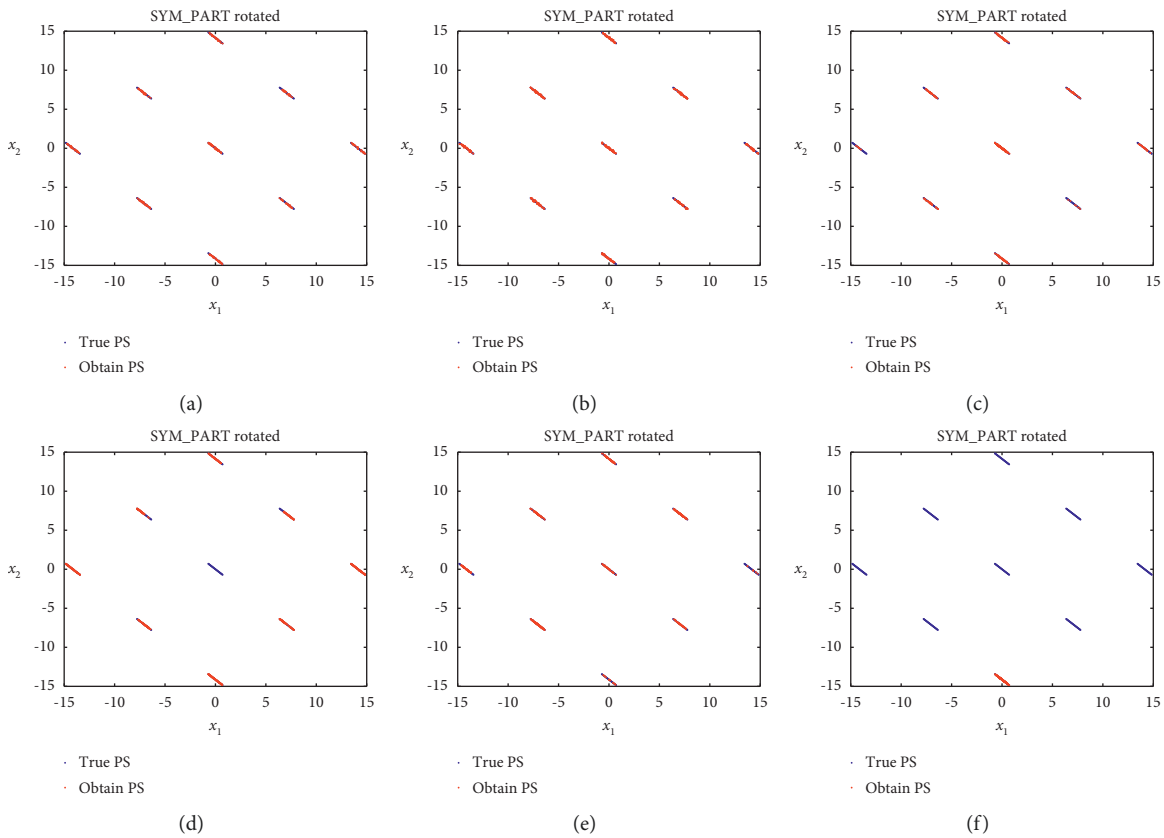


Figure 6: The population distributions of the true PSs and the final solutions obtained by different MMOEAs for SYM_PART rotated with nine equivalent PSs, (a) MOEA/D-HSA, (b) MO_Ring_PSO_SCD, (c) TriMOEA-TA&R, (d) DN-NSGA-II, (e) LORD, and (f) MOEA/D. Effectiveness analysis of MOEA/D-HSA.

Table 3: Average rankings of MOEA/D-HSA and MOEA/D-HSA variants by the Friedman test in *PSP*.

Algorithm	Rank
MOEA/D-SS	**1.5909**
SS-variant-1	2.4545
SS-variant-2	1.9545

Table 4: Average rankings of MOEA/D-HSA with different parameter settings by the Friedman test in *PSP*.

Algorithm	Rank
$t = 2$	3.5
$t = 3$	3.1818
$t = 4$	3.1818
$\boldsymbol{t = 5}$	**2.1364**
$t = 6$	3

Table 5: Normalized wall-clock time of six algorithms.

	MMOEA/D-HSA	MO_Ring_PSO_SCD	TriMOEA-TA&R	DN-NSGA-II	MOEA/D	LORD
MMF1	0.0056	0.0169(+)	**0.0045**(−)	0.0220(+)	0.0049(−)	1.0000(+)
MMF1_e	0.0111	0.0317(+)	**0.0046**(−)	0.0213(+)	0.0057(−)	1.0000(+)
MMF1_z	0.0051	0.0133(+)	**0.0036**(−)	0.0107(+)	0.0037(−)	1.0000(+)
MMF2	0.0071	0.0339(+)	**0.0039**(−)	0.0437(+)	0.0059(−)	1.0000(+)
MMF3	0.0071	0.0327(+)	**0.0039**(−)	0.0383(+)	0.0061(−)	1.0000(+)
MMF4	0.0040	0.0124(+)	**0.0027**(−)	0.0072(+)	0.0037(−)	1.0000(+)

Table 5: Continued.

	MMOEA/D-HSA	MO_Ring_PSO_SCD	TriMOEA-TA&R	DN-NSGA-II	MOEA/D	LORD
MMF5	0.0059	0.0181(+)	**0.0039**(−)	0.0219(+)	0.0049(−)	1.0000(+)
MMF6	0.0053	0.0168(+)	**0.0036**(−)	0.0204(+)	0.0047(−)	1.0000(+)
MMF7	0.0038	0.0115(+)	**0.0034**(−)	0.0093(+)	0.0036(=)	1.0000(+)
MMF8	0.0042	0.0160(+)	**0.0025**(−)	0.0088(+)	0.0041(=)	1.0000(+)
MMF9	0.0035	0.0102(+)	**0.0031**(−)	0.0043(+)	0.0028(−)	1.0000(+)
MMF10	0.0044	0.0107(+)	0.0029(−)	0.0048(+)	0.0028(−)	1.0000(+)
MMF11	0.0028	0.0088(+)	0.0028(=)	0.0034(+)	0.0026(=)	1.0000(+)
MMF12	0.0042	0.0171(+)	**0.0034**(−)	0.0080(+)	0.0037(−)	1.0000(+)
MMF13	0.0049	0.0126(+)	**0.0036**(−)	0.0050(+)	0.0040(−)	1.0000(+)
MMF14	0.0655	0.1534(+)	0.0500(−)	0.0603(−)	0.0259(−)	1.0000(+)
MMF14_a	0.0803	0.1813(+)	0.0453(−)	0.0762(−)	0.0309(−)	1.0000(+)
MMF15	0.0385	0.1573(+)	0.0535(+)	0.0612(+)	0.0262(−)	1.0000(+)
MMF15_a	0.0471	0.1925(+)	0.0618(+)	0.0791(+)	0.0342(−)	1.0000(+)
SYM_PART simple	0.0079	0.0422(+)	**0.0044**(−)	0.0172(+)	0.0061(−)	1.0000(+)
SYM_PART rotated	0.0103	0.0502(+)	**0.0037**(−)	0.0283(+)	0.0066(−)	1.0000(+)
Omni-test	0.0060	0.0240(+)	**0.0015**(−)	0.0102(+)	0.0042(−)	1.0000(+)
Sum up	+/ = /-	22/0/0	2/1/19	20/0/2	0/3/19	22/0/0

3.6. Analysis of Running Time. With the aim of comparing running time, all algorithms run 25 times on each test function, and the normalized wall-clock time of all algorithms is given in Table 5. As revealed in Table 5, TriMOEA-TA&R and MOEA/D have the shortest running time on 16 and 6 test functions, respectively. And, LORD takes the longest running time on all test functions. In addition, compared with the remaining algorithms, MMOEA/D-HSA achieves the shortest running time on most test functions. It can be concluded that the running time of MOEA/D-SS is not the shortest, but the difference is acceptable.

4. Conclusions

A simple but effective method MOEA/D-HSA is developed to tackle MMOPs. In MOEA/D-HSA, first, multiple individuals are assigned to the same weight vector to form a subpopulation to find equivalent PSs. Then, an environmental selection based on greedy selection is used to dynamically adjust the subpopulation size. Finally, a modified harmony search algorithm and elite learning strategy are utilized for balancing the diversity and convergence of the population. The experimental results prove that MOEA/D-HSA is superior to selected popular algorithms with respect to the IGDF and PSP values.

Besides, in this article, the dimension of the test problems is low. It is a meaningful study to develop MOEA/D-HSA on more complicated problems or the real applications [48, 49], such as the neural architecture search problems.

Data Availability

Data and code can be made available upon request to 915847903@qq.com.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This work was supported in part by the National Nature Science Foundation of China under Grant 61772391 and 62106186, in part by the Natural Science Basic Research Plan in Shaanxi Province of China under Grant 2022JQ-670, and in part by the Fundamental Research Funds for the Central Universities under Grant QTZX22047.

References

[1] F. Kudo, T. Yoshikawa, and T. Furuhashi, "A Study on Analysis of Design Variables in Pareto Solutions for Conceptual Design Optimization Problem of Hybrid Rocket Engine," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 2558–2562, New Orleans, LA, USA, June 2011.

[2] J. Michalek, R. Choudhary, and P. Papalambros, "Architectural layout design optimization," *Engineering Optimization*, vol. 34, no. 5, pp. 461–484, 2002.

[3] X. He, C. Fei, Y. Liu, K. Yang, and X. Ji, "Multi-objective Longitudinal Decision-Making for Autonomous Electric Vehicle: A Entropy-Constrained Reinforcement Learning Approach," in *Proceedings of the 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6, Rhodes, Greece, September 2020.

[4] G. Li, Q. Zhang, and Z. Wang, "Evolutionary Competitive Multitasking Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 26, 2022.

[5] X. Zheng, T. Gong, X. Li, and X. Lu, "Generalized scene classification from small-scale datasets with multitask learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, 2021.

[6] G. Li and Q. Zhang, "Multiple penalties and multiple local surrogates for expensive constrained optimization," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 769–778, 2021.

[7] G. Li, Q. Zhang, Q. Lin, and W. Gao, "A three-level radial basis function method for expensive optimization," *IEEE Transactions on Cybernetics*, vol. 99, pp. 1–12, 2021.

[8] L. Jinjian, A. Doniec, J. Boonaert et al., "A New Integrated Multimodal Multi-Criteria Route Planning Method in Bus Network Considering Passengers's Walking Behavior," in *Proceedings of the 2018 3rd IEEE International Conference On Intelligent Transportation Engineering (ICITE)*, pp. 13–17, Singapore, September 2018.

[9] Y. Han, D. Gong, Y. Jin, and Q. Pan, "Evolutionary multi-objective blocking lot-streaming flow shop scheduling with machine breakdowns," *IEEE Transactions on Cybernetics*, vol. 49, no. 1, pp. 184–197, 2019.

[10] O. Schütze, M. Vasile, and C. A. C. Coello, "Computing the set of epsilon-efficient solutions in multiobjective space mission design," *Journal of Aerospace Computing, Information, and Communication*, vol. 8, no. 3, pp. 53–70, 2011.

[11] X. Shan, Y. Li, X. Chen, Y. Wu, Y. He, and W. Li, "Investigations on internal ballistic characteristics of pasty propellant rocket engine," *International Journal of Aerospace Engineering*, vol. 2021, no. 8, pp. 1–14, Article ID 9952209, 2021.

[12] A. Jaszkiewicz, "On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 402–412, 2002.

[13] J. Liang, C. Yue, and B. Qu, "Multimodal Multi-Objective Optimization: A Preliminary Study," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 2454–2461, Vancouver, BC, Canada, July 2016.

[14] O. M. Shir, M. Preuss, B. Naujoks, and M. Emmerich, "Enhancing Decision Space Diversity in Evolutionary Multi-objective algorithms," in *Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization*, pp. 95–109, Nantes, France, April 2009.

[15] K. Deb and S. Tiwari, "Omni-optimizer: a generic evolutionary algorithm for single and multi-objective optimization," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1062–1087, 2008.

[16] C. Yue, B. Qu, and J. Liang, "A multiobjective particle swarm optimizer using ring topology for solving multimodal multiobjective problems," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 5, pp. 805–817, 2018.

[17] Y. Liu, H. Ishibuchi, G. G. Yen, Y. Nojima, and N. Masuyam, "Handling imbalance between convergence and diversity in the decision space in evolutionary multimodal multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 3, pp. 551–565, 2020.

[18] Y. Liu, G. G. Yen, and D. Gong, "A multimodal multiobjective evolutionary algorithm using two-archive and recombination strategies," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 4, pp. 660–674, 2019.

[19] Q. Fan and X. Yan, "Solving multimodal multiobjective problems through zoning search," *IEEE Transactions on Systems*, vol. 51, 2019.

[20] Q. Fan and O. K. Ersoy, "Zoning search with adaptive resource allocating method for balanced and imbalanced multimodal multi-objective optimization," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 6, pp. 1163–1176, 2021.

[21] Q. Lin, W. Lin, Z. Zhu, M. Gong, J. Li, and C. A. C. Coello, "Multimodal Multi-Objective Evolutionary Optimization with Dual Clustering in Decision and Objective Spaces," *IEEE Transactions on Evolutionary Computation*, vol. 25, 2020.

[22] K. Zhang, C. Shen, J. He, and G. G. Yen, "Knee based multimodal multi-objective evolutionary algorithm for decision making," *Information Sciences*, vol. 544, pp. 39–55, 2021.

[23] K. Zhang, M. S. Chen, X. Xu, and G. G. Yen, "Multi-objective evolution strategy for multimodal multi-objective optimization," *Applied Soft Computing*, vol. 101, 2021.

[24] K. Zhang, C. Shen, W. Hu, G. G. Yen, Z. Xu, and J. He, "Two-stage double niched evolution strategy for multimodal multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 99, p. 1, 2021.

[25] L. Ma, S. Cheng, and Y. Shi, "Enhancing learning efficiency of brain storm optimization via orthogonal learning design," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 11, pp. 6723-6724, 2020.

[26] X. Zheng, X. Chen, X. Lu, and B. Sun, "Unsupervised change detection by cross-resolution difference learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–16, 2021.

[27] Z. Li, L. Shi, C. Yue, Z. Shang, and B. Qu, "Differential evolution based on reinforcement learning with fitness ranking for solving multimodal multiobjective problems," *Swarm and Evolutionary Computation*, vol. 49, pp. 24–244, 2019.

[28] Q. Zhang and H. Li, "MOEA/D: a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2008.

[29] C. Hu and H. Ishibuchi, "Incorporation of a Decision Space Diversity Maintenance Mechanism into MOEA/D for Multi-Modal Multi-Objective Optimization," in *Proceedings of the Genetic And Evolutionary Computation Conference Companion*, pp. 1898–1901, Kyoto, Japan, July 2018.

[30] Y. Peng and H. Ishibuchi, "A Decomposition-Based Large-Scale Multimodal Multi-Objective Optimization Algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1–8, Glasgow, UK, July 2020.

[31] R. Tanabe and H. Ishibuchi, "A decomposition-based evolutionary algorithm for multi-modal multi-objective optimization," in *Proceedings of the International Conference on Parallel Problem Solving from Nature*, pp. 249–261, Coimbra, Portugal, September 2018.

[32] R. Tanabe and H. Ishibuchi, "A framework to handle multimodal multiobjective optimization in decomposition-based evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 4, pp. 720–734, 2020.

[33] M. Pal and S. Bandyopadhyay, "Decomposition in decision and objective space for multi-modal multi-objective optimization," *Swarm and Evolutionary Computation*, vol. 62, no. 1, 2021.

[34] W. H. Li, T. Zhang, R. Wang, and H. Ishibuchi, "Weighted Indicator Based Evolutionary Algorithm for Multimodal Multi-Objective Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 6, 2021.

[35] R. Tanabe and H. Ishibuchi, "A niching indicator-based multimodal many-objective optimizer," *Swarm and Evolutionary Computation*, vol. 49, pp. 134–146, 2019.

[36] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: nsga-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[37] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Proceedings of the International Conference on Parallel Problem Solving from Nature*, pp. 832–842, Birmingham, UK, September 2004.

[38] L. Li, G. G. Yen, A. Sahoo, L. Chang, and T. Gu, "On the estimation of pareto front and dimensional similarity in many-objective evolutionary algorithm," vol. 563, pp. 371–400, Information Sciences, 2021.

[39] B. Qu, G. Li, Q. Guo, L. Yan, X. Z. Chai, and Z. Q. Guo, "A Niching Multi-Objective harmony Search Algorithm for Multimodal Multi-Objective Problems," in *Proceedings of the 2019 IEEE Congress on Evolutionary Computation*, Wellington, New Zealand, June 2019.

[40] X. Li, K. Qin, B. Zeng, L. Gao, and L. Wang, "A dynamic parameter controlled harmony search algorithm for assembly sequence planning," *International Journal of Advanced Manufacturing Technology*, vol. 9, no. 2, pp. 3399–3411, 2017.

[41] B. Y. Qu, P. N. Suganthan, and S. Das, "A distance-based locally informed particle swarm model for multimodal optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 387–402, 2013.

[42] J. Liang, B. Qu, D. W. Gong, and C. T. Yue, "Problem Definitions and Evaluation Criteria for the Cec 2019 Special Session on Multimodal Multiobjective Optimization," *Computational Intelligence Laboratory*, Zhengzhou University, 2019.

[43] A. Aimin Zhou, Q. Yaochu Jin, and Y. Jin, "Approximating the set of Pareto-optimal solutions in both the decision and objective spaces by an estimation of distribution algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1167–1189, 2009.

[44] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[45] J. Liang, Q. Guo, C. Yue, B. Qu, and K. Yu, "A Self-Organizing Multiobjective Particle Swarm Optimization Algorithm for Multimodal Multiobjective Problems," in *Proceedings of the International Conference on Swarm Intelligence*, Shanghai, China, June 2018.

[46] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometric Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[47] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Publications of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1939.

[48] Z. Zhang, S. Liu, Y. Yang, and Y. Bai, "A link-adding strategy for improving robustness and traffic capacity in large-scale wireless sensor networks," *Cluster Computing*, vol. 22, no. 3, pp. 7687–7694, 2019.

[49] X. Zheng, B. Wang, X. Du, and X. Lu, "Mutual attention inception network for remote sensing visual question answering," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, 2021.

Hindawi
Discrete Dynamics in Nature and Society
Volume 2022, Article ID 9605189, 13 pages
https://doi.org/10.1155/2022/9605189

Research Article

Heuristic Algorithm for Cross-Platform Credit Risk Transmission Based on Hybrid Strategies

Zhang Xiaodong,[1] **Shen Hong**,[1] **Wang Tao**,[2] **and Li Yazhi**[3]

[1]*School of Information and Engineering, Nanjing Audit University, Nanjing 211185, Jiangsu, China*
[2]*Jusfoun Big Data Information Group Co., Ltd, Beijing 211185, China*
[3]*School of Software Engineering, Jinling Institute of Technology, NanJing 211169, Jiangsu, China*

Correspondence should be addressed to Shen Hong; shenhong@nau.edu.cn

Received 8 March 2022; Revised 1 April 2022; Accepted 7 April 2022; Published 25 May 2022

Academic Editor: Xueyi Wang

Credit risk transmission between cross-platforms is an important issue in the construction of a credit service system. The effect of credit risk transmission between credit entities (nodes) is analyzed in this paper. A heuristic algorithm based on hybrid strategies (HAHS) is proposed to find risk transmission paths and calculate the influence of nodes. Besides, a novel community model is applied to predict the credit risk areas in advance. In detail, the mathematical association structure between credit entities is firstly given in the algorithm, and the breadth first search algorithm is used to find the hierarchical nodes on the credit risk transmission paths. Then, the characteristics of credit risk transmission are analyzed, and the calculation methods of single-path and multipath influence are proposed. Finally, the credit entities are divided into communities based on a greedy strategy considering the characteristics of the credit entity association structure. The threshold control strategy is adopted to find global key nodes among all of the entities and local key nodes in communities, respectively, so as to realize the early warning of credit risk.

1. Introduction

In recent years, with the development of cloud computing and big data technology, online service has become the mainstream operation mode in all walks of life. With the convenient service of the Internet, all kinds of cross-platform users can obtain various production and operation services through a simple application process anytime and anywhere. However, credit risk is everywhere due to the virtuality of the network, the complexity of the production and operation process, and the dynamic change of user credit evaluation in the time interval. In fact, credit risk does not exist in isolation. The credit entity (node) changes under the influence of macroeconomics, market investment, financing environment, and other macro factors, which may lead to credit risk. On the other hand, the economic connection between enterprises caused by production cooperation and equity relationship may lead to risk transmission. The correlation between credit entities reflects the transmission mode and path of credit risk [1]. The description of the correlation between credit entities should reflect the complex connections between credit entities in social production and life. The transmission of credit risk is mainly related to the association structure and the correlation strength between credit entities. Many researchers have discussed this work from the perspective of theoretical and empirical analysis.

Daily banking practice shows that there may be contagion effects between portfolios, which has been clearly recognized through current supervision. Literature [2] describes a model that distinguishes default behavior in each portfolio and allows credit risk contagion between portfolios, including macroeconomic and financial factors. In addition, it also stimulates the multivariable scenario of portfolio credit risk. Literature [3] constructs the unconditional correlation network between listed financial institutions and comprehensively deconstructs the overall correlation of financial networks and the correlation

characteristics within and between departments through network analysis. Literature [4] analyzes the complex network theory and epidemiological thought to study the infection of the financial crisis on the global stock market. Through the causality method, the asymmetric impact between different markets can be reflected. Literature [5] applied the Diebold Yilmaz connectivity method to sovereign credit default swaps (SCDSs) to estimate the global network structure of sovereign credit risk. Literature [6] uses a factor model and elastic net shrinkage to model a high-dimensional network of European credit default swap (CDS) spreads. It also provides dynamic estimates of risk transmission, which is a useful tool for systemic risk monitoring. Literature [7] proposed a multilayer network model for credit risk assessment. The model takes into account the multiple connections between borrowers (such as geographical location and economic activities), allows explicit modeling of the interaction between related borrowers, and develops a multilayer personalized PageRank algorithm. Literature [8] constructs a network model of credit risk contagion in the interbank lending market based on time series. By the theoretical deduction and simulation method, how the contagion effects of credit risk accumulate and spread in the interbank market network is studied. In addition, it studies the evolution characteristics of credit risk contagion caused by the initial default of debt banks in the interbank market. In literature [9], many data samples are used to build an early warning model of Internet credit risk. The constructed model is trained and tested by BP neural network algorithm, and the genetic algorithm (GA) is used to optimize the neural network to improve the accuracy of early warning.

For the business activities of enterprises, literature [10] constructs a credit default estimation model for micro and small enterprises (MSEs) under the condition of changing information asymmetry. By relaxing the assumption that the bank can fully observe the customer's initial information, it constructs a theoretical model with practical application value. The model can accurately estimate the default probability of MSEs by quantitatively modeling the mechanism of default risk management and control. Literature [11] proposed an e-commerce credit risk evaluation method based on the language consensus model and established individual consensus measurement and group consensus measurement planning models to improve the consensus level of decision-making groups. Literature [12] studies the in-depth application of financial technology, which will reconstruct the internal relationship of enterprises in the supply chain, truly integrate small and medium-density enterprises into the network system of the supply chain, and turn the business behavior data of small and medium-density enterprises in the whole industrial chain ecology into "evaluable credit." Literature [13] studies credit risk contagion among affiliated enterprises by constructing the Markov chain and believes that credit risk contagion is the main cause of cluster default. Literature [14] considers that credit risk contagion between banks and firms is one of the important triggers of the financial crisis, and the credit linkage network is the way of systemic risk contagion triggered by external shocks. Literature [15] constructs a two-layer network model of credit risk contagion between the bank and corporate counterparties from the perspective that banks do not withdraw loans from enterprises by considering the influence of corporate credit defaults on their counterparties under the credit linkage. Literature [16] analyzes the contagion path of credit risk in Internet P2P lending. Based on complex network theory and the theory of infectious disease dynamics, the characteristics of Internet P2P lending development are combined to construct a SEIR model of credit risk transmission among Internet P2P lending platforms with a time lag, and the robustness of the model is analyzed and proven. Literature [17] proposes a two-stage hybrid model, credit data high-dimensional transformation model and graph-based neural network model, to enhance the prediction performance of credit risk. Literature [18] first analyzes the main factors affecting the performance of BSO and then proposes an orthogonal learning framework to improve its learning mechanism. In addition, a set of auxiliary transmission vectors with different characteristics are balanced through OD decision mechanism. Finally, the algorithm is verified on a set of benchmark tests and is used to solve the problem of quantitative association rule mining considering support, confidence, understandability, and netconf. For large-scale multiobjective optimization problems, an adaptive local decision variable analysis method based on decomposition is proposed in [19]. In this method, the guidance of the reference vector is incorporated into the analysis of control variables, and the adaptive strategy is used to optimize the decision variables.

In the current credit service mode of cross-platform, cross-domain, and cross-ecological, the sources of credit information are uneven, and credit risks are everywhere. There is no effective scheme for the transmission mode and influence calculating of credit risk between credit entities to meet the requirements of high real-time. To solve this problem, this paper analyzes the characteristics of credit risk transmission between cross-platform credit entities. Based on hybrid strategies, a heuristic algorithm is proposed. It finds the credit transmission path, constructs influence calculation models for both single-path and multipath transmission, and uses a greedy strategy to divide the credit entities into communities. In addition, threshold control strategies are applied to find global and local key nodes which can have a great impact on the associated credit entities and give the credit risk area in advance.

2. Problem Description

The cross-platform credit risk transmission problem considered in this paper is to find the credit risk transmission paths, compute the influence of credit risk on entities, and search for the key node sets from a set of n interrelated credit entities. The problem is analyzed for the purpose of improving the efficiency of credit risk prevention and reducing the possible risks caused by credit risk transmission.

This section firstly studies the association structure of credit entities and the credit risk transmission model, then

gives the influence calculation method when credit risk is transmitted through the associated entities, and finally puts forward the rules of community division.

2.1. Credit Entity Association Structure and Risk Transmission Model. As shown in Figure 1, in the complex credit entity association structure model, the importance of each credit entity (i.e., node) is related to the structure of the model and the correlation strength of the adjacency relationship between nodes. It is difficult to accurately measure the importance of nodes by a single factor.

In order to study the credit risk transmission path and influence on credit entities, n nodes in the association structure are transformed into a directed graph $G=(V,E)$, where V represents the set of nodes and E represents the set of edges. Each node in set V represents a credit entity, and each edge in set E represents the relationship between credit entities (such as parent-child relationship, equity relationship, supply relationship). The mathematical model of a given credit entity association structure is an n-order square matrix M which is composed of elements shown in the following formula:

$$d_{ij} = \begin{cases} w_{i,j} < i, j > \in E, \\ 0 < i, j > \notin E. \end{cases} \tag{1}$$

The weight $w_{i,j}\,(0 < w_{i,j} < 1)$ of each edge represents the correlation strength, and the value is given by experience according to the type of relationship between credit entities. Hence, the matrix M is shown in the following formula:

$$M = \begin{pmatrix} d_{11} & d_{12} & \dots & d_{1N} \\ d_{21} & d_{22} & \dots & d_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ d_{N1} & d_{N2} & \dots & d_{NN} \end{pmatrix}. \tag{2}$$

Existing empirical studies in credit-related fields have shown that the transmission of credit risk is mainly related to the association structure of credit entities and the correlation strength between credit entities. Based on the credit entity association structure model and some research conclusions from the existing literature on the social relationship network, this section models the credit risk transmission problem according to the following principle and method:

(1) Three-degree influence principle:

Fowler et al. [20] put forward the principle of three degrees of influence: nodes can not only affect neighbor nodes (one degree) but also affect neighbor nodes (two degrees) of neighbor nodes and even affect neighbor nodes (three degrees) of neighbor nodes. As long as they belong to a strong connection within three degrees, they are likely to cause risk transmission. If it is more than three degrees, the influence between nodes will be weakened to negligible or even disappear.

(2) Measurement method based on degree-centrality of topological network:

The problem of node centralization aims to study the centrality of nodes in the graph [21]. That is, the more frequently and closely connected with other nodes, the higher the centralization index, and nodes close to others with higher indexes generally have higher centralization indexes. The degree-centrality of the node reflects the influence of the current node in the whole credit entity association structure. This paper adopts the outdegree of node i to measure its degree centrality. The reason is that the higher the output degree of a node, the more nodes it affects and the greater its influence. This measurement strategy is also used to reduce the complexity of the algorithm.

2.2. Influence Calculation Method. When credit risk occurs to a credit entity, the transmission modes are mainly described as follows.

2.2.1. Single Path Transmission. This transmission mode reflects the multilevel transmission of credit risk among credit entities. Assuming that credit risk occurs to node i (called source node), its influence on itself is $\rho_{i,i} = 1$, and the transfer path from node i to node j is, $i \longrightarrow s_1 \longrightarrow s_2 \longrightarrow \cdots \longrightarrow s_k \longrightarrow j$, then the single-path influence calculation expression of node i on node j is shown as the following equation:

$$\rho_{i,j} = \rho_{i,i} \times w_{i,s_1} \times \cdots \times w_{s_{k-1},s_k} \times w_{s_k,j}. \tag{3}$$

Obviously, formula (2) ensures that the influence of credit risk on the transmission path is continuously reduced and reflects the buffer ability of each credit entity to credit risk. On the other hand, it also meets that the impact on the end node and the initial node of the path is positive correlation. As shown in Figure 2, the influence of nodes 1 on 2 is $\rho_{1,2} = 1 \times 0.3 = 0.3$, the influence on node 3 is $\rho_{1,3} = 1 \times 0.3 \times 0.4 = 0.12$, and the influence on node 4 is $\rho_{1,4} = 1 \times 0.3 \times 0.4 \times 0.3 = 0.036$.

2.2.2. Multipath Transmission. When credit risk occurs to the source node, it will be passed to the successor nodes through multiple paths, and some of these successor nodes may be affected by multiple predecessors at the same time. As shown in Figure 3, node 6 is affected by both nodes 3 and 5. It is assumed that the influence of node 3 on 6 is $\rho_{3,6}$, and the influence of node 5 on 6 is $\rho_{5,6}$. For node 6, in order to ensure that the influence of multipath transfer superposition is not lower than any influence of single-path, and the influence is less than or equal to 1, the calculation formula of multipath composite influence is shown as follows: $\rho_{*,6} = 1 - (1 - \rho_{3,6}) \times (1 - \rho_{5,6})$. Especially, $1 - \rho_{3,6}$ represents the degree of uncorrelation of node 3 on 6, and the similar to $1 - \rho_{5,6}$. The composite uncorrelation degree is obtained by multiplying, and the composite influence of multipath credit risk transmission is calculated.

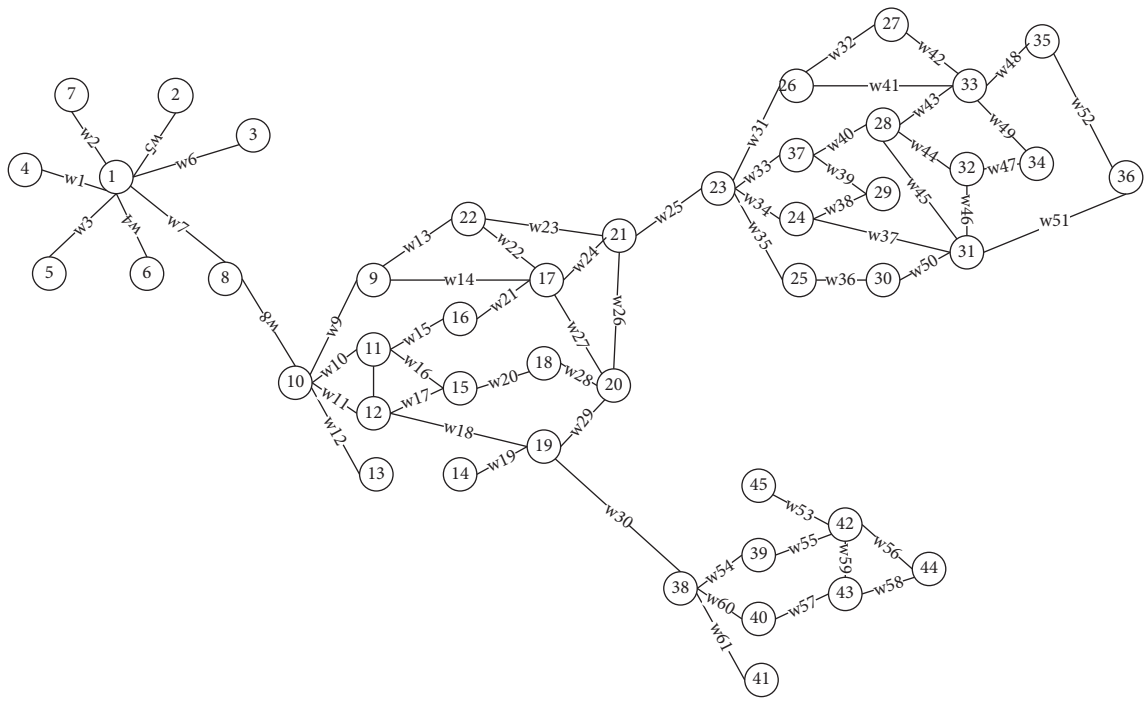


Figure 1: Credit entity association structure model.

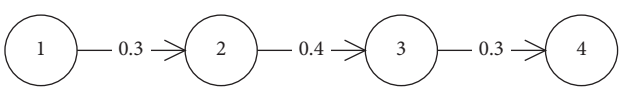


Figure 2: Continuous influence.

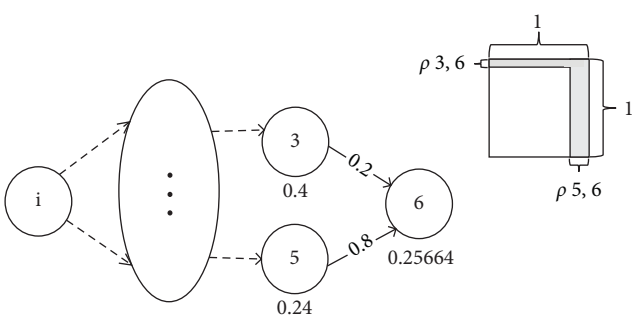


Figure 3: Influence superposition.

It can be seen from Figure 3, the value of $\rho_{i,3}$ is 0.4, and the value of $\rho_{i,5}$ is 0.24; thus, the influence of the source node i on node 6 through node 3 is $\rho_{3,6} = 0.4 \times 0.2 = 0.08$, and the influence on node 6 through node 5 is $\rho_{5,6} = 0.24 \times 0.8 = 0.192$. Therefore, the composite influence is obtained as follows: : $\rho_{*,6} = 1 - 1 - \rho_{3,6} \times 1 - \rho_{5,6} = 1 - (1 - 0.08) \times (1 - 0.192) = 0.25664$.

Based on the above description, the composite influence calculation model under multipath transmission is proposed. It can be described as follows, with the given credit entity association structure, if credit risk occurs to node *i*, it will pass through multiple-path to node *j*. The multipath composite influence calculation expression on node *j* is shown as follows:

$$\rho_{i,j} = 1 - \prod_{\forall <pre,j> \in E} 1 - \rho_{i,pre} \times w_{pre,j}, \tag{4}$$

where pre is any direct predecessor node of j, $\rho_{i,\text{pre}}$ is the influence of the source node i on node pre, and $w_{\text{pre},j}$ is the correlation strength from node pre to node j.

2.2.3. Circular Transmission. In the complex credit entity correlation structure, there may be a ring structure between many closely related entities. The transmission of credit risk is circular transmission in the ring structure, resulting in the circular calculation of influence. Therefore, this structure should be avoided. When calculating the influence of multipath credit risk transmission, this paper adopts the following preprocessing strategies to avoid the emergence of the ring: along with the direction of influence transmission, expand the successor adj of the current node i layer by layer according to the hierarchical structure. In the expansion process, if the node adj has appeared in the upper layer, it indicates that the node has already been affected at a higher level. According to the three-degree influence principle, the lower the level, the weaker the influence of the node. Therefore, the influence of the current node i on its successor adj can be ignored. Repeat the above layer-by-layer expansion operation until all nodes are traversed. Applying this strategy can avoid the ring on the delivery path in advance.

As the example shown in Figure 4, assuming that the source node with credit risk is node 1, it can be seen that nodes 3, 4, and 5 will form a ring structure. However, using the above strategy, when the influence is transmitted layer by layer, node 1 first affects nodes 2 and 4, node 2 then affects

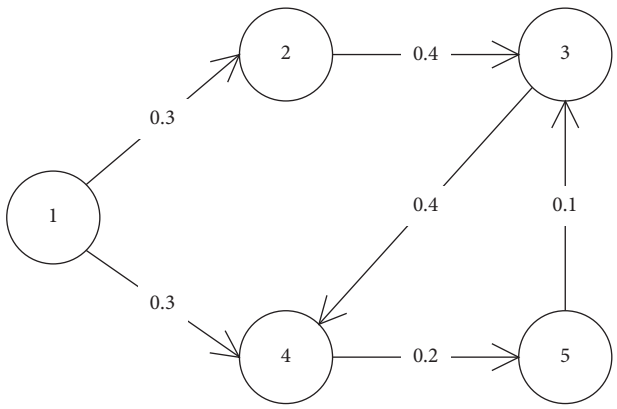


Figure 4: Circular transmission (before treatment).

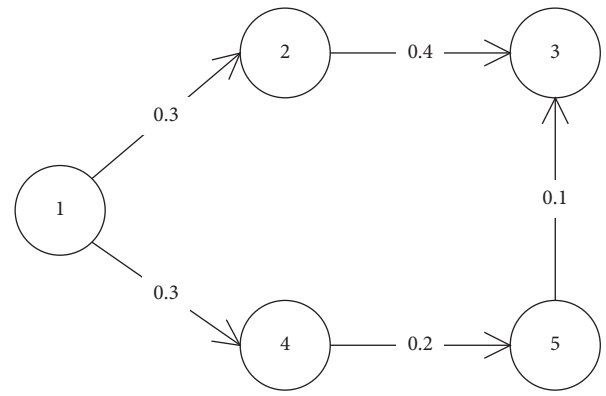


Figure 5: Circular transmission (after treatment).

node 3. Since node 4 has been affected by node 1, when node 3 is affected by node 2, the influence will no longer be transmitted to node 4. By applying this strategy to solve the credit risk transmission path of this example, the result is shown in Figure 5, which avoids the emergence of the ring structure mentioned above.

2.3. Division of Community. In the credit entity association structure, the correlation strength between nodes is determined by the relationship category between credit entities, and the subset of nodes with a strong association relationship may constitute a community. The internal connection of the community is close and influential, while the connection with external nodes is low-density. There may be multiple communities composed of different credit entities from the node set V. One node may form a special community, called an isolated community, which has a weak association with other communities.

One of the objectives of this paper is to find out the node sets from all of the credit entities. By forming different communities, the efficiency of credit risk supervision can be improved, and the possible harm of credit risk can be reduced. The method of community division in this paper is described as follows:

(1) Initial set P with all the nodes in V, t = 0

(2) If node set $P \neq \varnothing$, select node i with the maximal degree-centrality from P to establish a new community $C_{t+1} = \{i\}$, and remove node i from P

(3) For each successor adj of node i, if the degree-centrality of node adj is greater than or equal to the mean degree-centrality of node i and all its successors, thus $C_{t+1} = C_{t+1} \cup \{adj\}$, and remove adj from P

(4) Repeat (2)-(3) until $P = \varnothing$

3. Heuristic Algorithm Based on Hybrid Strategies

In this section, a heuristic algorithm based on a hybrid strategy (HAHS) is proposed for the considered problem. It aims to search the credit transmission paths, calculate the influence of nodes on the credit risk paths, divide credit entities into different communities, and find key nodes for each community. The proposed heuristic algorithm is depicted in Algorithm 1.

It mainly contains 6 steps with 4 subalgorithms. First, the credit entity association data is inputted and the mathematical model of the association structure is constructed. Next, a breadth first search algorithm (BFS) is used to search the credit risk transmission paths when the risk occurs to a credit entity. The influence calculation algorithm (Cal) is suggested to calculate the credit risk transmission impact through not only single-path but also multipaths. Global search algorithm (GS) is proposed to find global key nodes of the whole association structure. Last, the Community division and local key nodes searching algorithm (CDLS) is used to divide all of the credit entities into communities and find the local key nodes of each community.

The time complexity of the HAHS algorithm is $O(n^3)$, and the detailed analysis is given in Sections 3.1 to 3.4.

3.1. Breadth First Search Algorithm for Transmission Paths. In the credit entity association structure, when credit risk occurs to a node, it will be transmitted to the node's associated nodes, but the influence cannot be transmitted back. Otherwise, a ring will be generated. Meanwhile, the successors that the node can influence are its subordinate nodes. Thus, the credit risk transmission mode has the following characteristics:

(1) The impact of credit risk is a one-way transmission. That is, only the nodes at the upper level can affect the nodes at the lower level.

(2) The node at the lower level may be affected by one or more nodes at the upper level.

(3) There are no isolated nodes that neither affect nor be affected by other nodes.

As credit risk is transmitted layer by layer, the breadth first search algorithm is proposed for the transmission paths shown in Algorithm 2. It should be noted that the maximum value of the affected layer is 3 based on the three-degree influence principle, so the variable lev in algorithm 2 is less than 3.

```
Input: Credit entity association data
Output: 0
(1)  Input the credit entity association data
(2)  Construct the mathematical model of association structure: n-order square matrix M
(3)  Cal BFS () to find the credit risk transmission path
(4)  Cal Cal () to calculate influence for each node
(5)  Cal GS () to find global key nodes in all of the credit entities
(6)  Cal CDLS () to divide all of the credit entities into communities and find the local key nodes in each community
(7)  Return 0
```

Algorithm 1: Heuristic Algorithm based on Hybrid Strategy (HAHS).

```
Input: n-order square matrix M, Credit risk source node i, Θ = {i}
Output:Affected node set Θ
(1)  start = 0, end = |Θ|, lev = 0
(2)  While lev < 3
        For i = start to end
          Set current node a as the i th element in set Θ
          For j = a to n
            If d_aj ≠ 0 and j ∉ Θ
              Θ = Θ ∪ j
          Endif
        Endfor
      Endfor
     lev = lev + 1, start = end, end = |Θ|
     Endwhile
(3)  Return Θ
```

Algorithm 2: Breadth first search algorithm (BFS).

The time cost of BFS algorithm is mainly in Step 2, and its time complexity is O (n^2), so the time complexity of the BFS algorithm is O (n^2).

3.2. Influence Calculation Method. When the transmission paths are obtained by algorithm 2, the influence of each node in the path is then to be calculated. As shown in section 2.2, the ring of credit risk transmission no longer exists. There are only two situations to be considered: single-path transmission and multipath transmission. For the single-path transmission, the influence of each node is calculated by the single-path influence calculation expression described in formula (2). For the multipath transmission, the influence of each node is calculated by the multipath composite influence calculation expression described in formula (3). The implementation of the influence calculation is shown in Algorithm 3. If the influence of a credit entity exceeds the credit change threshold "T" which is set by the service supervision platform, the platform needs to change the entity's credit.

The time cost of the Cal algorithm is mainly in Step 2, and its time complexity is O (n^2); thus, the time complexity of the Cal algorithm is O (n^2).

3.3. Searching for Global Key Nodes. If a node has a great impact on other nodes in the whole domain, it is called a global key node. In order to give early warning to other nodes in advance to reduce the impact of credit risk, it is necessary to search for the global key nodes. A greedy strategy is given to select global key nodes among all of the credit entities. The basic idea of this method is to calculate the total influence of each node on other nodes and the number of nodes affected by the node and take those nodes that meet the defined threshold θ as global key nodes. The specific implementation is depicted in algorithm 4.

The time cost of GS algorithm is mainly in Step 2 and Step 3, and its time complexity is O (n), so the time complexity of the GS algorithm is O (n).

3.4. Community Division and Local Key Nodes Searching. The idea of finding the local key nodes in communities is similar to that of finding global key nodes among all of the credit entities. The difference is that all of the credit entities should first be divided into communities as described in Section 2.3, and then the key nodes in each community, i.e., local key nodes, are to be found. The specific implementation is shown in Algorithm 5. It produces a set S composed of several communities and a set Z containing sets of local key nodes for the communities.

```
Input: n – ordermatrixM, creditrisksourcenodei, affectednodesetΘ
Output: Ω = {⟨node, influence⟩}
(1)  Ω = ϕ
(2)  For each node a ∈ Θ
        If ρ_{i,a} ≠ 0
           continue;
        Endif
        Ψ = ϕ
        For j = 1 to a − 1
           If d_{j,a} ≠ 0
              Ψ = Ψ ∪ j
           Endif
        Endfor
        For each node p ∈ Ψ
           Recursively call this algorithm to calculate the influence of node p
        Endfor
        If |Ψ| = 1
           ρ_{i,a} = ρ_{i,pre} × w_{pre,a}
        Else
           ρ_{i,a} = 1 − ∏_{∀<pre,a>∈E} 1 − ρ_{i,pre} × w_{pre,a}
        Endif
        Ω = Ω ∪ {⟨a, ρ_{i,a}⟩}
     Endfor
(3)  Return Ω
```

Algorithm 3: Influence calculation method (Cal).

```
Input: n – ordermatrixM, thesetofΩ
Output: T = {global key nodes}
(1)  T = ϕ
(2)  For each node i in node set V
        Find the sum of influence Sum_i obtained by Algorithm 2
        Endfor
(3)  For each node i in node set V
           If Sum_i ≤ θ and deg_i ≤ θ
              T = T ∪ {i}
           Endif
        Endfor
(4)  Return T
```

Algorithm 4: Global Search (GS).

The time cost of the CDLS algorithm is mainly in Step 4, and its time complexity is O (n^3), so the time complexity of the CDLS algorithm is O (n^3).

4. Experiment

The proposed HAHS algorithm is written in Java language and runs on Intel (*R*) core (TM) i5-5200U CPU@ 2.2 GHz, 2.19 GHz, 8 GB RAM personal computer and MS Windows 10 operating system. The simulation experiment of credit risk transmission is first carried out, then a large number of instances with different sizes are tested and compared.

4.1. Simulation Experiment. This section takes the association structure with 20 credit entities as an example to conduct the simulation experiment of credit risk transmission.

After inputting the credit entity association data, the 20-order square matrix M is generated in the second step, and the result is shown in Table 1.

Assuming that credit risk occurs to node 1, i.e., node 1 is the source node. After the BFS algorithm is applied in the third step, the results of the credit risk transmission paths are obtained, shown in Figure 6. In this figure, it shows that the transmission paths of credit risk are expanded layer by layer. The first layer contains nodes 2, 3, 4, 5, 6, 8, and 10, the second layer contains nodes 7, 9, 11, 12, 13, 14, 15, 16, 17, and 18, the third layer is composed of nodes 19 and 20.

In Step 4, the Cal algorithm is used to compute the influence of each node on the above paths. The resulting key value pairs $\langle \mathrm{i}, \rho_{1,i} \rangle$ of the influence are as follows:

```
Input: G = (V, E), n − ordermatrix M,{<node, influence>}
Output: Community set S, Z = {O_1, O_2, ...} where O_i = {key nodes set for community i}
(1)  S = ∅, Z = ϕ
(2)  For each node i in node set V
       Calculate the centrality of node idepictedasdeg_i
     endfor
(3)  Repeat
         Select the node i with the largest degree of centrality
         C_i = ϕ, C_i = C_i ∪ {i}
         For each node j and d_ij ≠ 0
           If deg_j ≥ deg_i and deg_j ≥ ∑_{∀k,d_ik≠0} deg_k
             C_i = C_i ∪ {j}
         Endif
       Endfor
       S = S + C_i
     Until nodes in node set V are all divided
(4) For each community C_i
         For each node j in community set C_i
       Find the sum of influence (sum_i) generated by node j obtained by algorithm 2
       Endfor
       For each node j in community set C_i
         If sum_j ≤ θ and deg_j ≤ θ
           O_i = O_i ∪ {j}
           Endif
       Endfor
       Z = Z + O_i
     Endfor
(5)  Return S, Z;
```

Algorithm 5: Community division and local key nodes searching (CDLS).

Table 1: An instance with 20 nodes.

Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0.62	0.62	0.56	0.59	0.45	0	0.54	0	0.57	0	0	0	0	0	0	0	0	0	0
2	0	0	0.56	0.52	0.54	0.43	0	0	0.4	0.4	0.61	0	0	0	0	0	0	0	0	0
3	0	0	0	0.35	0	0.63	0.46	0.54	0.45	0	0.43	0.43	0	0	0	0	0	0	0	0
4	0	0	0	0	0.51	0.52	0.62	0	0.42	0.52	0	0	0.62	0.64	0	0	0	0	0	0
5	0	0	0	0	0	0	0.36	0	0	0.57	0.39	0.59	0.53	0.64	0	0.6	0	0	0	0
6	0	0	0	0	0	0	0.51	0.42	0	0.59	0.61	0	0.63	0	0	0.6	0.33	0	0	0
7	0	0	0	0	0	0	0	0.36	0.41	0.47	0	0	0.33	0	0.58	0	0.5	0	0.39	0
8	0	0	0	0	0	0	0	0	0.51	0.42	0	0.45	0.51	0.52	0.42	0	0	0.61	0	0
9	0	0	0	0	0	0	0	0	0	0.65	0.48	0.59	0.41	0.39	0	0.41	0	0.36	0	0
10	0	0	0	0	0	0	0	0	0	0	0.34	0.36	0	0.48	0.41	0.58	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0.65	0.59	0.43	0.33	0.46	0.52	0.65	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0.52	0.38	0.38	0.59	0.42	0.59	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0.35	0.61	0.62	0.44	0.45	0	0.62
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0.5	0.57
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.61	0.5	0.6	0.42	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.39	0.52	0	0.65
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.58	0.43
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.55
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.33
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

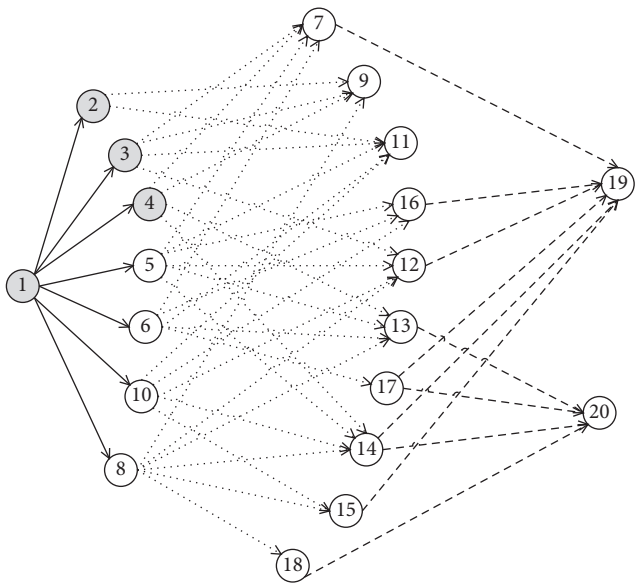


Figure 6: The transmission path of credit risk.

(1) The first layer: {<2,0.62>, <3,0.62>, <4, 0.56>, <5, 0.59>, <6, 0.45>, <8, 0.54>, <10, 0.57>}

(2) The second layer: {<7,0.717>, <9,0.700>, <11,0.795>, <12,0.712>, <13,0.767>, <14,0.791>, <15,0.407>, <16,0.684>, <17, 0.148>, <18,0.329>}

(3) The third layer: {<19, 0.809>, <20, 0.910>}

According to the GS algorithm in Step 5, the global key nodes are found, and these nodes include nodes 1, 2, 3, and 4. Once credit risk occurs to any of these nodes, it will have a significant impact on other nodes in the credit entity association structure.

Figure 7 shows the community division result based on the CDLS algorithm. It can be seen that the 20 credit entities are divided into five communities: A, B, C, *D*, and E. There are 12 nodes in community A, and nodes 1, 2, and 3 are the local key nodes of community A. There are 4 nodes in community B, and node 10 is the local key node of community B. There are 2 nodes in Community C, and node 17 is the local key node of community C. For the local key nodes of each community, once credit risk occurs to any one of these nodes, it will have an important impact on other nodes in its community. As nodes 18 and 20 have a weak influence on other nodes, each of them forms a community independently; i.e., they are isolated communities.

Through the above simulation experiment, the following conclusions can be obtained:

(1) Credit risk transmission is transmitted from near to far by layer diffusion

(2) Under the multipath composite influence, the influence of credit risk on a node does not consequentially weaken with the increase of hierarchy, and some outer nodes may be greatly influenced, leading to credit changes

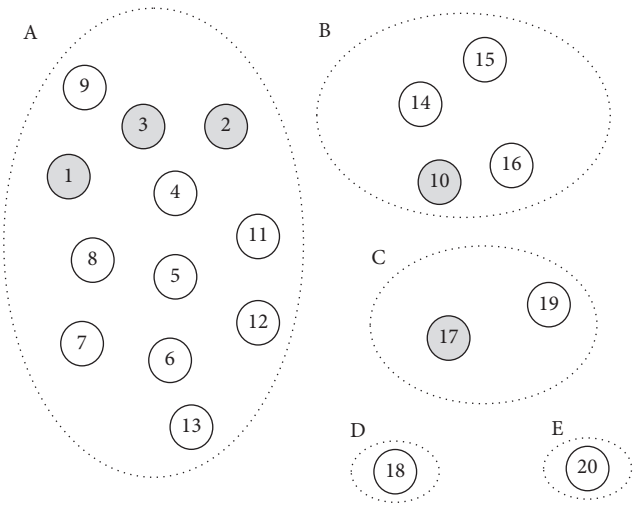


Figure 7: Division of community.

(3) Through the comparison between the global key nodes and the local key nodes, it can be found that a global key node is not necessarily a local key node and vice versa

4.2. Comparable Experiment. This section gives more comparative tests for node sets with different sizes and correlation densities.

4.2.1. Parameter Setting. In the credit entity association structure, if the nodes are closely related, that is, the number of associated nodes is much larger than the number of nonassociated nodes. The matrix M of the association structure is a high-density matrix; otherwise is a low-density matrix. In order to verify whether the algorithm has a significant impact on the node-set with different densities or not, this paper sets the association density property as 0.25, 0.45, and 0.65, which correspond to the low-density matrix, medium-density matrix, and high-density matrix, respectively.

In the following experiments, there are two factors, i.e., the sum influence of a node on its successors and the number of successors affected by the node, to determine whether a node can become a key node or not. Theoretically, if the sum influence of a node on its successors and the number of successors affected by the node are both greater than other nodes', the node can be taken as a key node. In the actual environment, the more key nodes, the greater the load of the service platform. Otherwise, if there are few key nodes, it is difficult to find the entities with credit risk in time. Experiments show that if the nodes with influence greater than the mean are taken as key nodes, the number of them accounts for about 30% of the total number of nodes. According to the Pareto principle, this paper finally selects the intersection of 20% of the nodes that meet the above two factors, respectively, as key nodes. Therefore, the threshold $''\theta''$ in this study is set to 0.2.

Table 2: Experimental results (low-density matrix).

Problem size ∗ number of instances	Number of global key nodes	Total number of local key nodes	Number of communities	Number of isolated communities
60 ∗ 10	10.6	13.5	11.2	6.1
90 ∗ 10	16.6	20.6	11.1	5.2
120 ∗ 10	21.7	26	11.7	6.1
150 ∗ 10	27	31.7	13.1	7.6
180 ∗ 10	32.1	38.5	12.7	6.3
210 ∗ 10	37.8	44.5	12.9	6.4
240 ∗ 10	43.5	51.4	12.3	5.3

Table 3: Experimental results (medium-density matrix).

Problem size∗number of instances	Number of global key nodes	Total number of local key nodes	Number of communities	Number of isolated communities
60 ∗ 10	10.8	14.1	7.9	3.1
90 ∗ 10	17.2	20.3	8.4	3.2
120 ∗ 10	23.2	26.6	9.2	3.7
150 ∗ 10	29.2	32.5	9.9	4.2
180 ∗ 10	34.9	38.7	9.4	3.4
210 ∗ 10	41	45	10.3	3.9
240 ∗ 10	46	51.6	10.7	3.5

Table 4: Experimental results (high-density matrix).

Problem size ∗ number of instances	Number of global key nodes	Total number of local key nodes	Number of communities	Number of isolated communities
60 ∗ 10	11.8	14	7.5	2.9
90 ∗ 10	17.5	20.7	7.9	2.5
120 ∗ 10	23.3	26.8	8.8	2.8
150 ∗ 10	29.5	32.8	9.3	3.2
180 ∗ 10	35.5	39	8.6	2.4
210 ∗ 10	41.4	45.6	9.3	2.4
240 ∗ 10	47.4	51.8	9.5	2.2

4.2.2. Experimental Results and Conclusions. This section randomly generates instances using the method of literature [22]. For each kind of density, the test data is divided into 7 groups, the problem size in each group are 60, 90, 120, 150, 180, 210, and 240, respectively, and each group generates 10 instances.

The experimental results of association structure with the low-density matrix are shown in Table 2.

The experimental results of association structure with the medium-density matrix are shown in Table 3.

The experimental results of association structure with the high-density matrix are shown in Table 4.

From the above experimental results, some characteristics of the considered problem can be concluded.

(1) For any instance, the total number of local key nodes is greater than the number of global key nodes. This is because the significance of local key nodes is that when credit risk occurs to a credit entity, its impact will spread violently within the community. In the global scope, the influence of a key node in one community is relatively weak or even has no influence on nodes in other communities. Therefore, the total number of local key nodes is greater than that of global key nodes.

(2) For instances with the same density, although the node size is increasing, the number of communities and isolated communities is growing slowly. This is because although the number of nodes has increased, the number of nodes in each community has also increased due to the improvement of the degree of association between nodes rather than forming more communities. Similarly, the number of isolated nodes increases slowly due to the improvement of the degree of association between nodes.

(3) For instances with different densities, it can be seen from the comparison of the above three tables: with the increase of matrix density, i.e., node correlation, the number of communities and the average number of isolated communities gradually decrease. This is because the closer the degree of association, the closer the relationship between nodes, and with a higher degree of mutual influence, the number of communities decreases gradually, and the

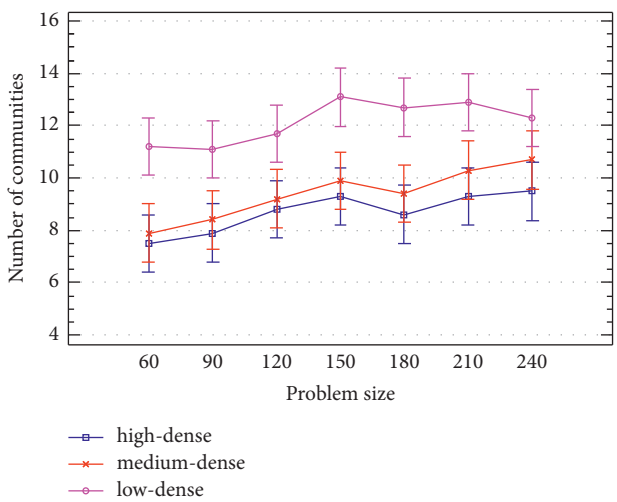


Figure 8: Comparison of the number of communities in the 95% Tukey HSD confidence interval.

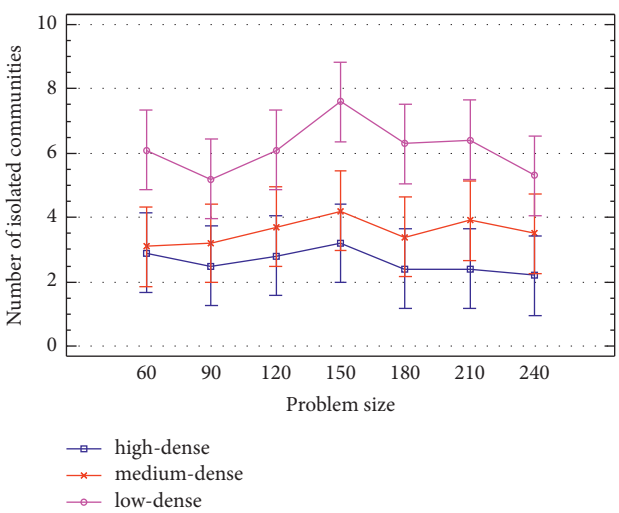


Figure 9: Comparison of the number of isolated communities in the 95% Tukey HSD confidence interval.

probability of forming isolated nodes also decreases gradually.

4.2.3. Further Analysis and Discussion. In order to better analyze the effectiveness of the algorithm, this section uses StatGraphics to statistically analyze the experimental results generated in the previous section. As mentioned in the above section, the experimental results come from 210 instances, which, respectively, describe the association structure matrix obtained under different problem sizes and different densities. The statistical analysis of the experimental results contains the number of communities, the number of isolated communities, the number of global key nodes, and the total number of community key nodes, as shown in Figure 8 to Figure 11.

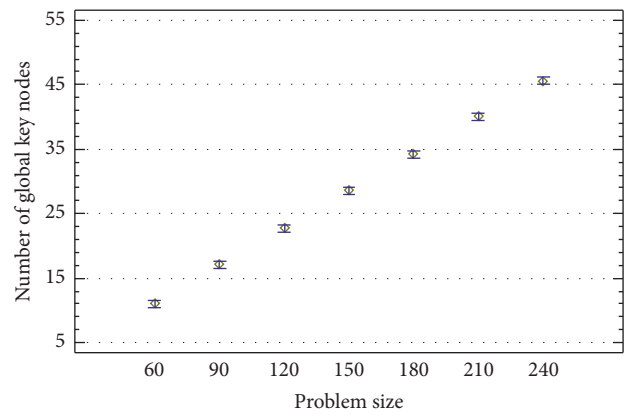


Figure 10: Comparison of the number of global key nodes in 95% Tukey HSD confidence interval.

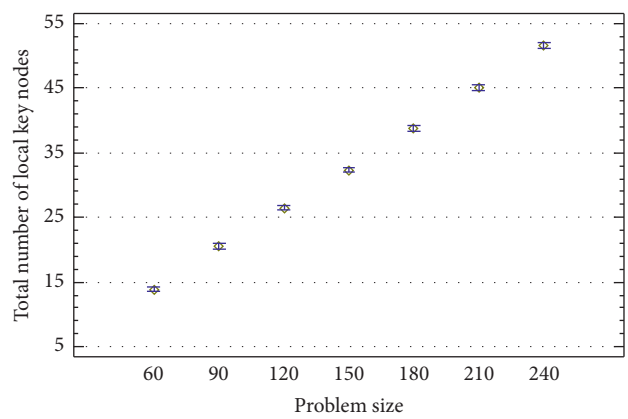


Figure 11: Comparison of the total number of local key nodes in 95% Tukey HSD confidence interval.

As shown in Figure 8, with the increase in the size of the problem, the total number of communities found shows a slow upward trend in the 95% Tukey HSD confidence interval, which is consistent with the second conclusion in the previous section. In addition, the higher the association density, the fewer communities are found, and vice versa. The reason is that the higher the association density is, the closer the credit entities are connected, and these closely connected credit entities are put into the same community, so fewer communities are obtained.

As shown in Figure 9, with the increase in the problem size, the total number of isolated communities found fluctuated up and down within the 95% Tukey HSD confidence interval, and there was no obvious upward trend. This statistical result is similar to the third conclusion in the previous section. Besides, the lower the association density, the more isolated communities are found, and vice versa. This is because the lower the association density, the more sparse the connections between credit entities, and these credit entities are more likely to form isolated communities.

In order to better realize the early warning of credit risk, this paper sets 20% as a threshold to select the qualified

credit entities as the key nodes, so the number of these key nodes increases linearly with the problem size. It can also be seen from Figures 10 and 11 that, with the increase of the problem size, the number of global key nodes found by the GS algorithm shows an obvious upward trend, so as to the number of local key nodes in the community found by CDLS algorithm.

5. Conclusion

This paper studies credit risk transmission of cross-platform credit services. Aiming at the considered problem, a heuristic algorithm based on hybrid strategies named HAHS is proposed. After modeling the association structure of credit entities, the characteristics of credit risk transmission are analyzed, and influence calculation methods of credit risk are proposed for single-path and multipath transmission. Finally, a greedy strategy is used for community division, and a threshold control strategy is applied to find global and local key nodes. The results of the simulation experiment show the effectiveness of the algorithm. Through comparative experiments, the characteristics of credit risk transmission are concluded, which further shows the algorithm can be used to solve the problem of cross-platform credit risk transmission.

Abbreviations

HAHS: Heuristic algorithm based on hybrid strategy
BFS: Breadth first search algorithm
Cal: Influence calculation algorithm
GS: Global search
CDLS: Community division and local key nodes searching.

Data Availability

No datasets were generated or analyzed during the current study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Key R&D Program of China (Grant no. 2019YFB1404602) and Open Project for Young Teachers in School of Information Engineering, Nanjing Audit University (Grant no. XG202102). The authors would like to thank Jiangsu Key Laboratory of Audit Information Engineering for their support and anyone who supported the publication of this paper.

References

[1] F. X. Diebold and K. Yılmaz, "On the network topology of variance decompositions: measuring the connectedness of financial firms," *Journal of Econometrics*, vol. 182, no. 1, pp. 119–134, 2014.

[2] A. Uquillas and R. Tonato, "Inter-portfolio credit risk contagion including macroeconomic and financial factors: a case study for Ecuador," *Economic Analysis and Policy*, vol. 73, pp. 299–320, 2022.

[3] Z. Li, Q. Liang, and X. F. Tu, "Research on the relevance of listed financial institutions in China -- based on network analysis," *Journal of Financial Research*, vol. 8, pp. 95–110, 2016.

[4] K. D. Wang, X. B. Pang, and S. S. Wang, "Research on the contagion of financial crisis to global stock market: based on complex network analysis method," *World Economy Studies*, vol. 4, pp. 28–39, 2018.

[5] G. Bostanci and K. Yilmaz, "How connected is the global sovereign credit risk network?" *Journal of Banking & Finance*, vol. 113, Article ID 105761, 2020.

[6] C. Gross and P. L. Siklos, "Analyzing credit risk transmission to the nonfinancial sector in Europe: a network approach," *Journal of Applied Econometrics*, vol. 35, no. 1, pp. 61–81, 2020.

[7] M. Óskarsdóttir and C. Bravo, "Multilayer network analysis for improved credit risk prediction," *Omega*, vol. 105, Article ID 102520, 2021.

[8] T. Chen, Y. Wang, Q. Zeng, and J. Luo, "Network model of credit risk contagion in the interbank market by considering bank runs and the fire sale of external assets," *Physica A: Statistical Mechanics and its Applications*, vol. 542, Article ID 123006, 2020.

[9] G. Du, Z. Liu, and H. Lu, "Application of innovative risk early warning mode under big data technology in Internet credit financial risk assessment," *Journal of Computational and Applied Mathematics*, vol. 386, Article ID 113260, 2021.

[10] T. Chen and S. Wang, "Incomplete information model of credit default of micro and small enterprises," *International Journal of Finance & Economics*, pp. 1–19, 2021.

[11] P. Wu, L. G. Zhou, and H. Y. Chen, "E-commerce credit risk evaluation method based on language consensus model," *Control and Decision*, vol. 36, pp. 1465–1471, 2021.

[12] K. S. Leung and Y. K. Kwok, "Counterparty risk for credit default swaps: Markov chain interacting intensities model with stochastic intensity," *Asia-Pacific Financial Markets*, vol. 16, no. 3, pp. 169–181, 2009.

[13] D. Petrone and V. Latora, "A dynamic approach merging network theory and credit risk techniques to assess systemic risk in financial networks," *Scientific Reports*, vol. 8, no. 1, pp. 5561–61, 2018.

[14] P. Mu, T. Chen, K. Pan, and M. Liu, "A network evolution model of credit risk contagion between banks and enterprises based on agent-based model," *Journal of Mathematics*, vol. 2021, pp. 1–12, Article ID 6593218, 2021.

[15] T. Chen, Q. Yang, Y. Wang, and S. Wang, "Double-layer network model of bank-enterprise counterparty credit risk contagion," *Complexity*, vol. 2020, Article ID 3690848, 25 pages, 2020.

[16] C. Zhao, M. Li, J. Wang, and S. Ma, "The mechanism of credit risk contagion among internet p2p lending platforms based on a SEIR model with time-lag," *Research in International Business and Finance*, vol. 57, Article ID 101407, 2021.

[17] J. Liu, S. Zhang, and H. Fan, "A two-stage hybrid credit risk prediction model based on XGBoost and graph-based deep neural network," *Expert Systems with Applications*, vol. 195, Article ID 116624, 2022.

[18] L. Ma, S. Cheng, and Y. Shi, "Enhancing learning efficiency of brain storm optimization via orthogonal learning design," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 11, pp. 6723–6742, 2021.

[19] L. Ma, M. Huang, S. Yang, R. Wang, and X. Wang, "An adaptive localized decision variable analysis approach to

large-scale multiobjective and many-objective optimization," *IEEE Transactions on Cybernetics*, vol. 99, pp. 1–13, 2021.

[20] J. H. Fowler and N. A. Christakis, "Dynamic spread of happiness in a large social network: longitudinal analysis over 20 years in the framingham heart study," *British Medical Journal*, vol. 3, pp. 1–31, 2009.

[21] L. Wang and Q. Q. Zhang, "Centralization of complex networks," *Complex Systems and Complexity Science*, vol. 3, no. 1, pp. 13–20, 2006.

[22] E. Demeulemeester, M. Vanhoucke, and W. Herroelen, "Rangen: a random activity-on-the-node network generator," *Journal of Scheduling*, vol. 6, no. 1, pp. 17–38, 2003.