Received October 23, 2020, accepted November 2, 2020, date of publication November 4, 2020, date of current version November 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3035899

An Agglomerative Greedy Brain Storm Optimization Algorithm for Solving the TSP

CHANGYOU WU AND XISONG FU
School of Management Science and Engineering, Shandong Institute of Business and Technology, Yantai 264005, China
Corresponding author: Changyou Wu (wuchangyou_81@163.com)

This work was supported by the National Natural Science Foundation of China under Grant 41601593.

ABSTRACT The brain storm optimization algorithm(BSO) is a population based metaheuristic algorithm inspried by the human conferring process that was proposed in 2010. Since its first implementation, BSO has been widely used in various fields. In this article, we propose an agglomerative greedy brain storm optimization algorithm (AG-BSO) to solve classical traveling salesman problem(TSP). Due to the low accuracy and slow convergence speed of current heuristic algorithms when solving TSP, this article consider four improvement strategies for basic BSO. First, a greedy algorithm is introduced to ensure the diversity of the population. Second, hierarchical clustering is used in place of the k-means clustering algorithm in standard BSO to eliminate the noise sensitivity of the original BSO algorithm when solving TSP. Exchange rules for the individuals in the population individuals were introduced to improve the efficiency of the algorithm. Finally, a heuristic crossover operator is used to update the individuals. In addition, the AG-BSO algorithm is compared with the genetic algorithm (GA), particle swarm optimization (PSO), the simulated annealing(SA) and the ant colony optimization (ACO) on standard TSP data sets for performance testing. We also compare it with a recently improved version of the BSO algorithm. The simulations show the encouraging results that AG-BSO greatly improved the solution accuracy, optimization speed and robustness.

INDEX TERMS Brain storm optimization algorithm, traveling salesman problem, hierarchical clustering, optimization algorithm, combinatorial optimization.

I. INTRODUCTION

In the past two decades, swarm intelligence(SI) algorithms have become highly influential. They mainly simulate the group behavior of insects, birds, fish and other populations. The main properties of an SI optimization algorithm are inspired by the social behaviors of a natural population. In an SI algorithm, each individual in the population represents a solution in the search space. SI algorithms have been extensively and effectively used to solve realistic problems, but the generalization ability for specific scenarios is poor. Therefore, more effective algorithms and search strategies have been studied. Scholars have developed many novel SI algorithms, such as particle swarm optimization(PSO) [1], ant colony optimization(ACO) [2], gray wolf optimization algorithm(GWO) [3], firefly algorithm(FA) [4], whale optimization algorithm(WOA) [5] and so on [6], [7].

In 2011, Shi [8] proposed a new swarm intelligent optimization algorithm at the second international conference on SI, named brain storm optimization(BSO), which simulated the thought process of human creative problem solving. This algorithm is also known as the intellectual incentive method. Compared with other intelligent optimization algorithms, BSO has the advantages of a compact mathematical model, simple operation, clear process, fast convergence speed and high optimization efficiency. Therefore, it is considered to be a very promising method, and it has been favored by many researchers and widely applied in practical optimization problems in different fields such as power systems [9]–[13], machine learning [14]–[19], combinatorial optimization problems [20]–[22] and image processing [23]–[25] and prediction [15], [26], [27].

In recent years, BSO has attracted attention due to its unique and excellent performance for solving complex and high-dimensional large-scale optimization problems. A large number of scholars have paid increasing attention to BSO and have conducted in-depth research on it. According to the algorithm mechanism and application background, the research on the BSO algorithm can be divided into the following categories: (1) improving the clustering method of

The associate editor coordinating the review of this manuscript and approving it for publication was Norbert Herencsar.

BSO [28]–[36]; (2) improving the new individual generation strategy [34], [37]–[44]; (3) applying the research on BSO [9], [12], [15], [18], [38], [45]–[47]. The improvement and research of these algorithms from different directions not only improve the optimization performance of BSO but also promote the healthy development of BSO theory and applications.

The clustering method of the BSO algorithm directly affects the convergence speed of the algorithm and is a major factor affecting the performance of BSO. Therefore, many scholars have focused their research on the choice of the clustering methods of BSO. The purpose of the clustering method in the BSO algorithm is to converge the solution to a certain area. Different clustering algorithms can be used in BSO. In the basic BSO algorithm. The k-means clustering algorithm is used. However, this original clustering strategy has been replaced by increasing number of convergence methods. In 2015, Shi [30] introduced a new convergence operation that replaced the k-means clustering method in the basic BSO algorithm; therefore, a large reduction in the computation time can be obtained. In 2016, Chen *et al.* [29] proposed an improved affinity propagation (AP) clustering method and an enhanced creation strategy for the structural information of single or multiple clusters to adaptively change the number of clusters during the search process. In 2018, Dash *et al.* [31] introduced k-means++ technology to improve the BSO algorithm, which solves the problem of the slow convergence of the algorithm by using a random probability decision in the river formation dynamics scheme to select the best clustering centroid for population generation. In 2018, Duan *et al.* [32] introduced a new clustering method based on metric distance into the basic BSO, proposed metric distance brainstorm optimization (MDBSO), and applied the improved algorithm. In 2018, Li *et al.* [33] proposed an improved BSO. In the improved algorithm, the original clustering strategy is modified, and a random grouping scheme is introduced to shrink the computational cost and maintain the diversity of the population. In 2019, Cao and Wang [35] proposed active learning brain storm optimization(ALBSO) to enhance the performance of the original BSO. The simulation results show that the performance of ALBSO has been significantly improved.

In the research on the individual generation strategy of BSO, in 2017, El-Abd [40] described a global-best version combined with per-variable updates to improve the performance of BSO. At the same time, the proposed algorithm incorporated a reinitialization scheme that was triggered by the current state of the population. In 2018, Li *et al.* [41] proposed a BSO algorithm with multi-information interactions (MIIBSO) that addresses the issue of untimely convergence for complex problems and balances exploration and exploitation. In 2018, Papa *et al.* [42] introduced a variant of the basic BSO. To increase the diversity of the population, this method uses different transfer functions to map real-valued solutions to Boolean hypercubes. In 2018, Song *et al.* [34] proposed a BSO that used simplified individual combinations and improved step functions to update the individuals. In addition, a quantum behavior mechanism was introduced into the improved BSO, and a quantum-behaved individual strategy with periodic learning behavior was developed. In 2019, Yu *et al.* [44] proposed distance-based diversity and fitness-based diversity to improve the diversity of the BSO population and to adapt the algorithm parameters. In 2020, Guo *et al.* [43] proposed a grid-based multiple objective BSO with a hybrid mutation operation, which drew on the idea of Cauchy and chaotic mutation operators and generated new individuals with wide diversity. In 2020, Sun *et al.* [37] proposed a novel BSO (RMBSO) in view of the fact that the original BSO tends to stagnate in the exploitation phase. The algorithm used a slight relaxation selection and the creation of thought sets based on multiple populations to improve the performance of BSO for global optimization problems with multiple landscapes.

In the application research on BSO, in 2016, Wang *et al.* [15] combined principal component analysis (PCA) and BSO for stock price prediction. In addition, the improved BSO algorithm was used to search for the best parameters for v-support vector regression(v-SVR). In 2018, Xu *et al.* [48] proposed an improved BSO based on prior knowledge(DBSO), and applied it to solve traveling salesman problems(TSPs). In 2018, Ke [45] used BSO to solve the cumulative capacitated vehicle routing problem which aims to minimize the sum of arrival times for customers. In 2018, Xiong *et al.* [12] proposed an improved BSO and applied it to solve fault section diagnosis (FSD) problems in power systems. In 2019, Palanikkumar and Priya [18] proposed an improved BSO and applied it to reliably manage medical information. In 2019, Hao *et al.* [47] proposed a hybrid BSO and applied it to solve distributed hybrid flowshop scheduling problems. In 2019, Chen *et al.* [46] introduced an improved BSO with a composite index, and applied the algorithm to solve the optimization problem in a hybrid renewable energy system. In 2019, Revathi *et al.* [49] proposed a hybrid BSO based whale optimization algorithm(BSO-WOA) for user privacy data protection. In 2020, Peng *et al.* [38] proposed a novel BSO, namely, the multicluster adaptive brain storm optimization (MCaBSO) algorithm, and applied the novel BSO to obtain the optimal combination of services based on the quality of service (QoS).In 2020, Narmatha *et al.* [25] proposed the fuzzy brain-storm optimization algorithm(FBSO) for medical image segmentation and classification. Experimental results showed that the proposed FBSO was efficient and robust, and significantly reduced the segmentation time. In 2020, Cervantes-Castillo and Mezura-Montes [21] described the enhancement of the BSO algorithm with a special operator and applied it to solve constrained numerical optimization problems.

In summary, a large number of scholars have contributed to the improvement of BSO which has proven to have value in realistic applications in various fields. Although scholars have made some progress in the theory and application of BSO, there are still some key issues. The enhancement of the

BSO algorithm proposed in [29], [30], [34], and [43] involve modifying only the clustering method and the strategy for generating the population. However, most BSO algorithms still tend to fall into local optima when solving complex optimization problems. As mentioned above, BSO is rarely applied to combinatorial optimization problems such as the TSP. To address this gap in current research, this article applies the improved BSO algorithm to the solution of the TSP. The TSP is proven to have NP-hard characteristics. The wide range ofvariants of the TSP encountered in the real life has made this problem a popular focus for researchers; however, solving the TSP quickly and effectively remains a considerable challenge. In recent years, an increasing number of algorithms have been used to solve the TSP [48], [50]–[52], and this type of problem has also been used to test the performance of algorithms. It should be noted that the literature listed in this section represents only a small part of the related work. Due to the large and growing number of relevant studies, summarizing all related work would be a difficult task. Therefore, readers who wish to learn more about the possible applications of BSO are referred to the article published by Cheng *et al.* [53]. On the other hand, for a further understanding of the TSP and related solution techniques, interested readers are recommended to consult the work introduced in [54] and [55]. A good BSO algorithm for soling TSP should have the following characteristics: (1) The individual update strategies should make full use of information on the fitness of the current population and the number of algorithm iterations. (2) When the algorithm is running, the loop statement should be reduced to increase the solution speed of the algorithm. (3) A balance between exploration and development should be achieved throughout the iterative search process. (4) For either a small-scale or large-scale TSP instances, the algorithm should be able to converge to the global optimal solution with high accuracy. Based on the above considerations, this article proposes an agglomerative greedy BSO algorithm (AG-BSO).

The main contributions of this work to research on the TSP are as follows:

- We propose a greedy BSO algorithm based on agglomerative hierarchical clustering. AG-BSO offers three main improvements over the basic BSO algorithm:
  1) We replace the original k-means clustering algorithm with an agglomerative clustering algorithm, thereby eliminating the influence of 'noise' and the sensitivity of boundary data.
  2) A greedy algorithm is introduced for population initialization.
  3) Exchange rules and a heuristic crossover operator are adopted for updating the individuals in the population.
- The AG-BSO algorithm and other classic algorithms proposed in the literature have been used to calculate results for the TSPLIB test set. The AG-BSO algorithm is found to be superior to the other algorithms in terms of solution time and solution accuracy.
- We compare the proposed AG-BSO algorithm with other improved BSO algorithms that have recently been presented and show that the proposed AG-BSO algorithm has great advantages in terms of solution quality.

The rest of this article is structured as follows: a description of the TSP and an introduction to the BSO algorithm introduction are given in Section II. In section III, the proposed improved BSO for solving the TSP is introduced in detail. The experiments performed are described in Section IV. Finally, a summary of the paper with conclusions and directions for future improvement, is presented in Section V.

II. THE BASIC PROBLEM DESCRIPTION

A. TRAVELING SALESMAN PROBLEM

The TSP is the most famous and widely studied problem in the history of computer science and operations research. Like a great deal of other combinatorial optimization and routing problems, the TSP is considered NP-hard. It is also a tool for testing algorithm performance. The TSP can be simply described as follows: a salesperson needs to visit n cities to sell products, and the salesperson can visit each city only once, randomly starting from one city and finally returning to the same city, such that the total distance of the selected route is the shortest pobsible. The mathematical model of the TSP is formulated as follows:

$$C = (c_1, c_2, \cdots, c_n), \quad 1 \leqslant c_i \leqslant n \tag{1}$$

$$F(c_i) = \sum_{i=1}^{n-1} D(c_i, c_{i+1}) + D(c_n, c_1) \tag{2}$$

where i is the number of the i-th city, with C represents the set of all city numbers; $D(c_i, c_{i+1})$ represents the distance between two cities and $D(c_i, c_{i+1}) = D(c_{i+1}, c_i)$. The goal of the problem is to find a Hamiltonian ring subject to the given constraints such that the value of Eq.(2) is minimized.

In this article, we use the well-known path representation to encode the TSP solution. Accordingly, each solution is coded as an arrangement of numbers [56]. The order in which the numbers are arranged indicates the order in which the salesman visits each city, and the Euclidean distance is used to calculate the distance between each pair of nodes.

B. THE PRINCIPLE OF BRAIN STORM OPTIMIZATION

BSO as an algorithm that gradually reduces the search space, has attracted the attention of researchers around the world because of its high convergence speed and solution accuracy. BSO is usually combined with individual clustering and mutation mechanisms, and the algorithm relies on two main operators: convergence operator and divergence operator. The core of the algorithm when solving a problem consists of the classification, selection and individual updating(mutation) of the solution scheme. The details are as follows:

- Clustering strategy: The original BSO algorithm uses the k-means clustering to cluster similar individuals into k clusters and then uses an artificially specified individual with the optimal fitness function value as the center of each cluster. To avoid falling into a local optimum, a new individual will also be randomly generated to replace one of the cluster centers.

- Selection strategy: A selection strategy is used to maintain better solutions among all individuals. After each new individual is generated, the selection strategy retains better solutions, while the clustering strategy and generation strategy are applied to add new solutions to the species to maintain the diversity of the entire population.
- Mutation strategy: there are four main mechanisms for mutation in BSO:
  1) Add a random disturbance to a random class center to generate a new individual;
  2) Randomly select an individual in a random class and add a random disturbance to generate a new individual.
  3) Randomly combine two class centers, and add random disturbances to generate new individuals.
  4) Randomly fuse two random individuals in two classes, and add random disturbances to generate new individuals.

The probability of each cluster center being selected in the above 4 mechanisms is shown in Eq.(3):

$$M_i = \frac{|K_i|}{Q} \tag{3}$$

where $|K_i|$ represents the size of population i in cluster K, and Q is the number of clusters. Gaussian mutation is used to generate new individuals in BSO, and the mutation operator is as follows:

$$X_{new}^d = X_{se}^d + \xi * N(0, 1)_d \tag{4}$$

where X_{select}^d is a new a-dimensional individual and X_{new}^d is the set of selected d-dimensional individuals. $N(0, 1)_d$ is a Gaussian distribution with mean 0 and variance 1, and ξ is a coefficient that weights the contribution of the Gaussian random value which is calculated by Eq.(5).

$$\xi = \log\mathrm{sig}\left(\left(0.5 * MaxIter - CurrentIter\right)/k\right) * R(0, 1) \tag{5}$$

where *MaxIter* and *CurrentIter* are the set maximum number of iterations and the current number of iterations, respectively. $\log sig()$ is a sigmoid transfer function. k is the coefficient of the slope of the adjustment function needed to balance the convergence rate of the algorithm, and $R(0, 1)$ is a random number in the interval [0, 1].

Algorithm 1 presents the corresponding the pseudocode of this procedure [8]. The implementation of the BSO algorithm is a simple, consisting of the following steps:

1) The population is initialized.
2) The individuals in the population are evaluated and clustered.
3) New individuals are randomly generated.
4) The population and cluster centers are updated;

Algorithm 1 Pseudocode of the Basic BSO

```
1: Randomly generate $N$ individuals and calculate their fitness.
2: $gen = 1$( Record the number of iterations).
3: while $gen < MaxIter$ do
4:     Use k-means clustering algorithm to divide $N$ individuals into $K$ clusters.
5:     Record the best individual as the cluster center.
6:     if $random(0, 1) < p = 0.2$ then
7:         Cluster center is replaced by randomly generated individuals.
8:     end if
9:     for $i = 1$ to $N$ do
10:        if $random(0, 1) < p\_one = 0.8$ then
11:            Randomly select a cluster.
12:            if $random(0, 1) < p\_one\_center = 0.4$ then
13:                Determine cluster center $X_{select}$.
14:            else
15:                Randomly select other individuals.
16:            end if
17:        else
18:            Randomly generate two different clusters $k1$ and $k2$.
19:            if $random(0, 1) < p\_two\_center = 0.5$ then
20:                Choose a combination of two cluster centers as $X_{select}$.
21:            else
22:                Randomly select two individual combinations as $X_{select}$.
23:            end if
24:            Update the individual $X_{select}$ by Eq.(4) and Eq.(5).

25:            Compare new and old individuals
26:        end if
27:    end for
28: end while
```

5) If the maximum number of iterations has been reached, the optimal individual is output; otherwise, the algorithm returns to 2).

III. IMPROVED BSO FOR THE TSP

A. INITIALIZATION OF THE TSP

The improved BSO is built on the basic BSO, and the population is initialized through a greedy algorithm [57] and a randomly generated population strategy based on prior knowledge. When the number of nodes in the city is small, the randomly generated population strategy is used to generate the initial population; otherwise, the greedy algorithm is used. In accordance with Meeran's [58] research, most individuals in the population are generated based on prior knowledge, whereas only a small percentage are randomly generated, such that mean that the updated population is representative and the algorithm has good converges

Algorithm 2 Pseudocode of Heuristic Crossover Operator

```
Input: parent_one, parent_two, dist
Output: child_one, child_two, start_c
1: for $i = 1$ to $n$ (Individuals in the parent) do
2:     Initialize child_one,child_two.
3:     Randomly generate $n$ individuals, generate a random number as the starting point for offspring, set $start = 6$.
4:     Update child_one, Find the right_one and right_two cities of the parent individual 6.
5:     if dist (right_one, 6) < dist (rigth_two, 6) then
6:         Update the element at the second position of child_one to right_one.
7:     else
8:         Update the element at the second position of child_one to right_two
9:     end if
10:    Delete city 6 from the parent and update the parent.
11:    Continue the above loop until one element.
12:    Similarly, update Child_two.
13: end for
```

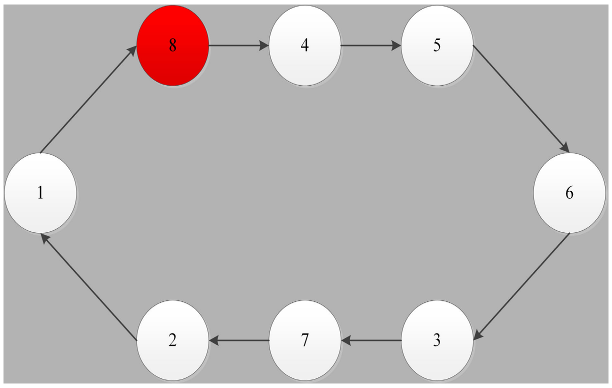


FIGURE 1. City sequence.

properties, as. This has also been indicated through experiments. Brief descriptions of the greedy algorithm and the randomly population generation strategy for updating the TSP population are provided as follows:

1) RANDOM POPULATION GENERATION STRATEGY

The generation of the initial population is the primary problem to be addressed when solving the TSP [59]. In this article, when the number of city nodes is fewer than or equal to 150, a set of values is randomly generated. Each true value represents the order in which the salesman travels [60]. For example, consider a randomly generated sequence containing 8 true values, as shown in Fig.1. The salesman starts at city 8, visits cities 4, 5, and so on until city 1, and finally returns to city 8 from city 1 to complete the cycle.

2) GREEDY ALGORITHM FOR POPULATION INITIALIZATION

When the number of city nodes is greater than 150, the greedy algorithm is used to generate the initial population to reach the best result quickly.The greedy algorithm starts by randomly selecting of one city as the starting city, adding it to the new population, and selecting the optimal population based on a mechanism for removing the worst case. Then, among all cities not included in the population, the city closest to the current city according to the Euclidean distance is calculated to update the current city. The above operations are repeated until all cities are included in the new population, thereby achieving population diversity [61].

B. SOLUTION CLUSTERING

The purpose of solution clustering is to make the solution converge to a small region. Various clustering algorithms can be used in the BSO algorithm [29], [53]. In the original BSO algorithm, the basic k-means clustering algorithm is used. However, due to the weak robustness of the k-means algorithm, it can easily fall into local optima, and the cluster centers have a great impact on the clustering results. Therefore, this article proposes a hierarchical clustering algorithm for population clustering in BSO for the first time.

The most obvious advantage of hierarchical clustering compared with partitioned clustering is that it reduces the chain effect [62]. In addition, a hierarchical clustering algorithm can be used to cluster data sets at different scales (levels). Hierarchical clustering algorithms may be agglomerative or divisive, depending on whether the hierarchical divisions are determined in the "bottom-up" or "top-down" [63]; a brief description of the process is shown in Fig.2. In this article the former type of algorithm is used to cluster the TSP population. For a given data set $L = (l_1, l_2, \cdots, l_n)$, the specific steps implementation of the agglomerative hierarchical clustering algorithm are as follows:

1) In the initial stage, each sample in the data set L is regarded as its own class $l_i, i = 1, 2, \cdots n$.
2) The matrix D of distance between each pair of samples in the data set L is calculated;
3) Based on the calculated distance matrix D, first, the pair of samples separated by the smallest distance is found, the smallest distance between samples is stored, and the two samples are merged into a new class; Second, the distance is recalculated. Finally, the distance matrix D is updated according to the Ward linkage method, for which the calculation formula is as follows:

$$D(r,s) = \sqrt{\frac{2J_rJ_s}{J_r+J_s}}\|\bar{y}_r - \bar{y}_s\|_2 \tag{6}$$

where J_r and J_s represent the number of samples in cluster r and cluster s, $\bar{y_r}$ and $\bar{y_s}$ represent the cluster center of cluster r and cluster s, and the symbol $\|\|_2$ represents the calculated Euclidean distance.

Based on the operational principles of agglomerative hierarchical clustering, it can be inferred that the time complexity of the algorithm is $O\left(n^3\right)$.This is because two different classes are merged each time, it is necessary to traverse the distance matrix to search for the minimum distance, which

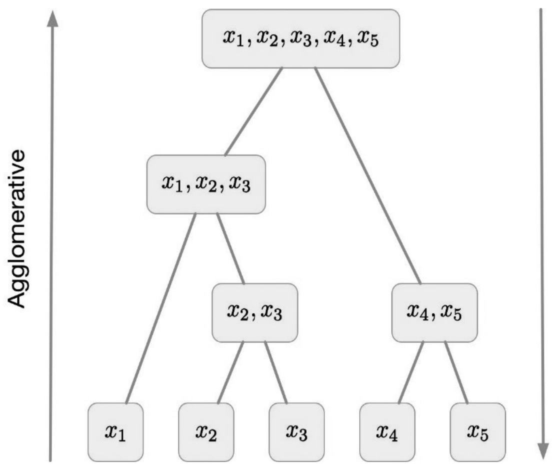


FIGURE 2. **Agglomerative and divisive process.**

is a process with time complexity $O\left(n^2\right)$, and finally, all classes are combined. For each cluster, this operation needs to be performed $n-1$ times. Experimental results indicate that the noise-resistance of this algorithm and the shapes of the resulting clusters are superior to those in the case of those of k-means clustering. Therefore, the agglomerative hierarchical clustering can improve the performance of the BSO algorithm while maintaining its efficiency.

C. GENERATION OF NEW INDIVIDUALS

The core of the BSO algorithm is the mutation and selection of new individuals. As in other intelligent algorithms, intergroup and intragroup interactions are considered [64]. By maintaining the diversity of the population, the BSO algorithm can efficiently find the optimal solution when solving the TSP. In this article, operators for exchanging individuals and heuristic crossover are used to generate individuals.

1) EXCHANGE INDIVIDUALS

To increase the diversity of the population, an exchange rule is adopted to update the population in the initial stage of individual cluster update process [65]. This exchange rule is illustrated by the example below.

Suppose that 8 cities are generated randomly and that the current solution is:

1	2	3	4	5	6	7	8

We randomly select two positions, 2 and 5, and exchange the individuals in these two positions; then, the solution is updated to:

1	5	3	4	2	6	7	8

2) HEURISTIC CROSSOVER OPERATOR

Considering the complexity of the TSP and referring to the prior knowledge of Li *et al.*, a heuristic crossover operator is used in this article to generate offspring from parents with outstanding genetic information to accelerate the algorithm's optimization speed [66]. We choose parent 1 and parent 2 from the initial population and perform crossover between them by means of the heuristic crossover operator to generate child 1 and child 2. Algorithm 2 presents the pseudocode of the heuristic crossover operator.

The flow of the improved BSO algorithm with hierarchical clustering is shown in Fig.3.

TABLE 1. **Algorithm related parameter settings.**

Related parameters	BSO	AG-BSO
Population: MM	100	100
Maximum number of iterations: $MaxIter$	600,1000	600,1000
Number of clusters: $cluster\_num$	5	5
Replace cluster center probability: $p\_replace$	0.4	0.3
Choose a clustering probabilities: p_one	0.5	0.6
Choose two clustering probabilitie: p_two	0.5	0.4
Choose a cluster center: p_one_center	0.4	0.45
Select two cluster centers:p_two_center	0.45	0.5

TABLE 2. **Results of the BSO and BSO1, BSO2 and BSO3 on four TSP datasets.**

Algorithm	Index	ulysses22	eil51	berlin52	pr226
BSO	MeanV	75.31	431.91	7544.36	82174.13
	MeanT	15.05	24.97	12.48	187.33
	Dev	0	1.39	0.03	2.25
BSO1	MeanV	77.04	436.02	7553.21	82352.07
	MeanT	13.05	20.43	10.93	120.13
	Dev	1.8	2.35	0.14	2.46
BSO2	MeanV	75.3	428.87	7543.33	81242.39
	MeanT	25.38	37.63	24.65	211.47
	Dev	0	0.67	0.02	1.09
BSO3	MeanV	75.31	435.03	7544.36	81243.72
	MeanT	14.97	25.32	17.5	87.12
	Dev	0	2.12	0.03	1.09

IV. EXPERIMENTAL STUDIES

In this section, experiments conducted on the TSP to test the improved BSO algorithm will be introduced in detail. First, we analyze the parameters related to the algorithm and the components of the improved BSO algorithm. For the TSP data sets, the 28 instances used in this article come from the TSPLIB benchmark [67]. The number of city nodes in these 28 instances ranges from 22 to 1291; the number in the name of each instance indicates the number of nodes it contains. For each instance, the algorithm introduced in this article was independently run 30 times. To evaluate the performance of AG-BSO, we compared it with the following advanced intelligent optimization methods: (1) traditional intelligent algorithms such as GA [68], SA [69], ACO [2], and PSO [1]; (2) the improved BSO algorithm [48], [70]; and (3) other improved intelligent algorithms [71]–[74]. The simulation experiments were run on a computer equipped with an Intel Core i5-5200 processor, and the program was created in the Windows 10 environment.

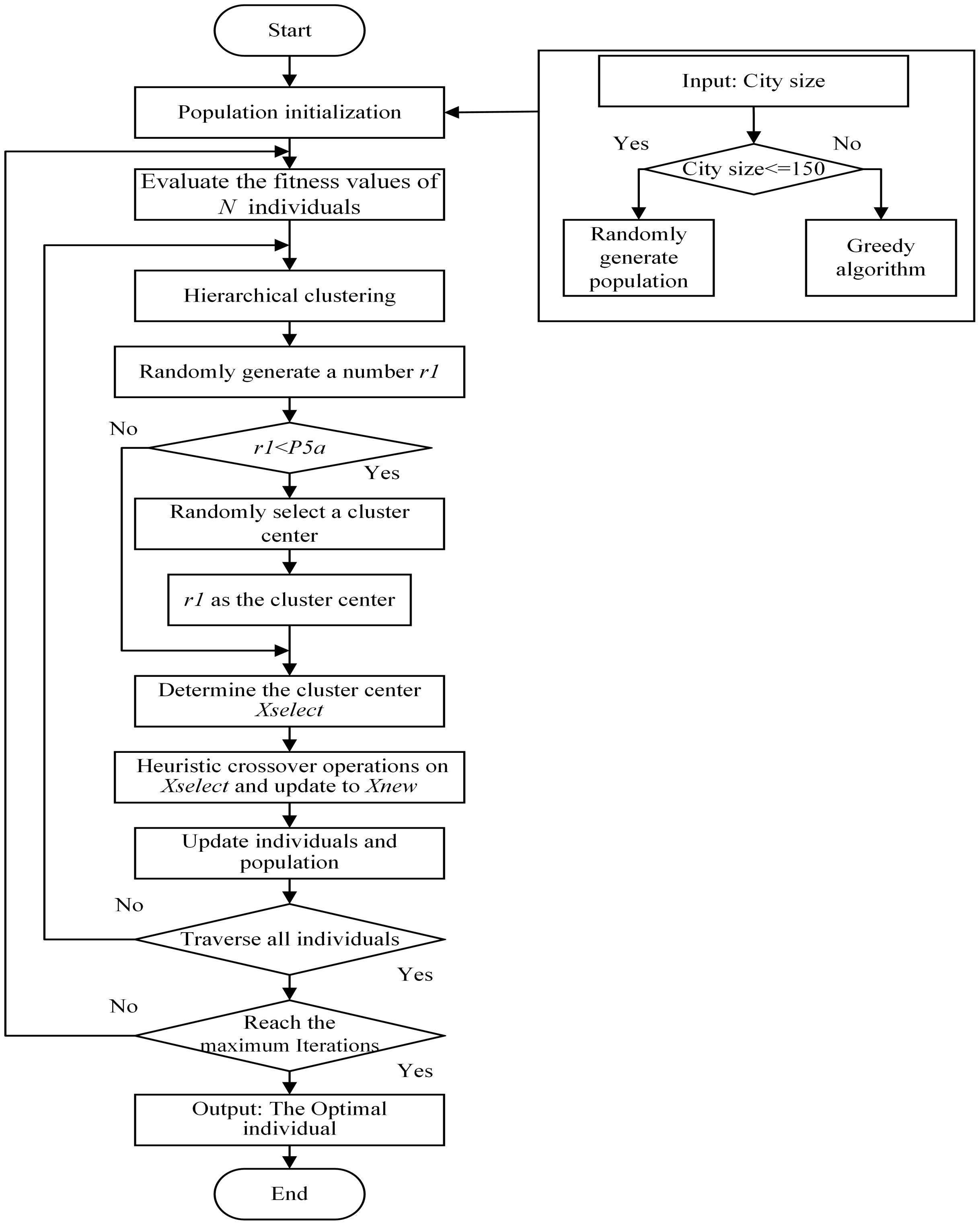

Start
Population initialization
Evaluate the fitness values of N individuals
Hierarchical clustering
Randomly generate a number r1
r1<P5a
No
Yes
Randomly select a cluster center
r1 as the cluster center
Determine the cluster center Xselect
Heuristic crossover operations on Xselect and update to Xnew
Update individuals and population
Traverse all individuals
No
Yes
Reach the maximum Iterations
No
Yes
Output: The Optimal individual
End
Input: City size
City size<=150
Yes
No
Randomly generate population
Greedy algorithm

FIGURE 3. The AG-BSO algorithm.

A. PARAMETER SETTINGS AND COMPONENT TESTING

The adjustment of the parameters is crucial to the performance of intelligent optimization algorithms. The parameter settings used for the original BSO algorithm and other algorithms in the literature were adopted from the related literature [8], [38], [41], [48], [71]. After in-depth discussion, the parameters of the AG-BSO algorithm were determined through repeated tests and revisions. To ensure the rigor of this work, the relevant algorithm parameters are listed in TABLE.1.

TABLE 3. Results of the proposed BSO algorithm and the basic BSO algorithm.

Instance		AG-BSO			BSO		
Name	Optima	Best	Dev(%)	Time(s)	Best	Dev(%)	Time(s)
ulysses22	75.67	**75.24**	0	9.05	75.31	0	15.05
att48	33522	33538.34	0.05	6.54	34920.15	4.17	11.37
eil51	426	428.58	0.61	12.19	431.91	1.39	24.97
berlin52	7542	**7542**	0	8.48	7544.36	0.03	12.48
st70	675	678	0.44	10.26	684.41	1.39	14.8
pr76	108159	108170.3	0.01	10.13	110816.35	2.46	15.29
eil76	538	540.69	0.5	9.32	555.98	3.34	12.94
rat99	1211	**1211**	0	11.4	1248.9	3.13	15.98
kroA100	21282	**21070.09**	0	11.88	21616.02	1.57	15.61
kroB100	22140	22152	0.05	11.5	22629.86	2.21	16.91
kroC100	20749	**20749**	0	11.92	20931.15	0.88	16.18
eil101	629	633	0.64	11.75	658.26	4.65	15.89
ch130	6110.86	6115.25	0.07	26.69	6233.18	2	67.11
ch150	6528	**6528**	0	17.5	6742.82	3.29	22.02
d198	15780	15951.29	1.08	74.85	16339.32	3.54	106.16
kroA200	29368	29507.35	0.47	49.31	30651.85	4.37	58.02
kroB200	29437	29678.92	0.82	64.37	30358.6	3.13	110.66
pr226	80369	80961.17	0.74	66.63	82174.13	2.25	187.33
gil262	2378	2394.98	0.71	82.1	2478.85	4.24	202.77
a280	2579	2583.12	0.16	83.82	2642.12	2.45	216.72
lin318	42090	43023.73	2.22	184.73	45687	8.55	245.31
fl417	11861	11936.25	0.63	257.88	12821.92	8.1	350.37
pr439	107217	113074.47	5.46	86.09	114774.9	7.05	102.31
pcb442	50778	52375.32	3.15	80.04	53268	4.9	96.58
d493	35002	36470.63	4.19	314.13	37568.59	7.33	437.01
rat575	6773	6929.25	2.31	373.28	7066.02	4.32	613.31
d657	48912	49204.51	0.59	445.99	51870.36	6.05	609.57
d1291	50801	51983.28	2.32	1321.59	53642.59	5.59	1768.56

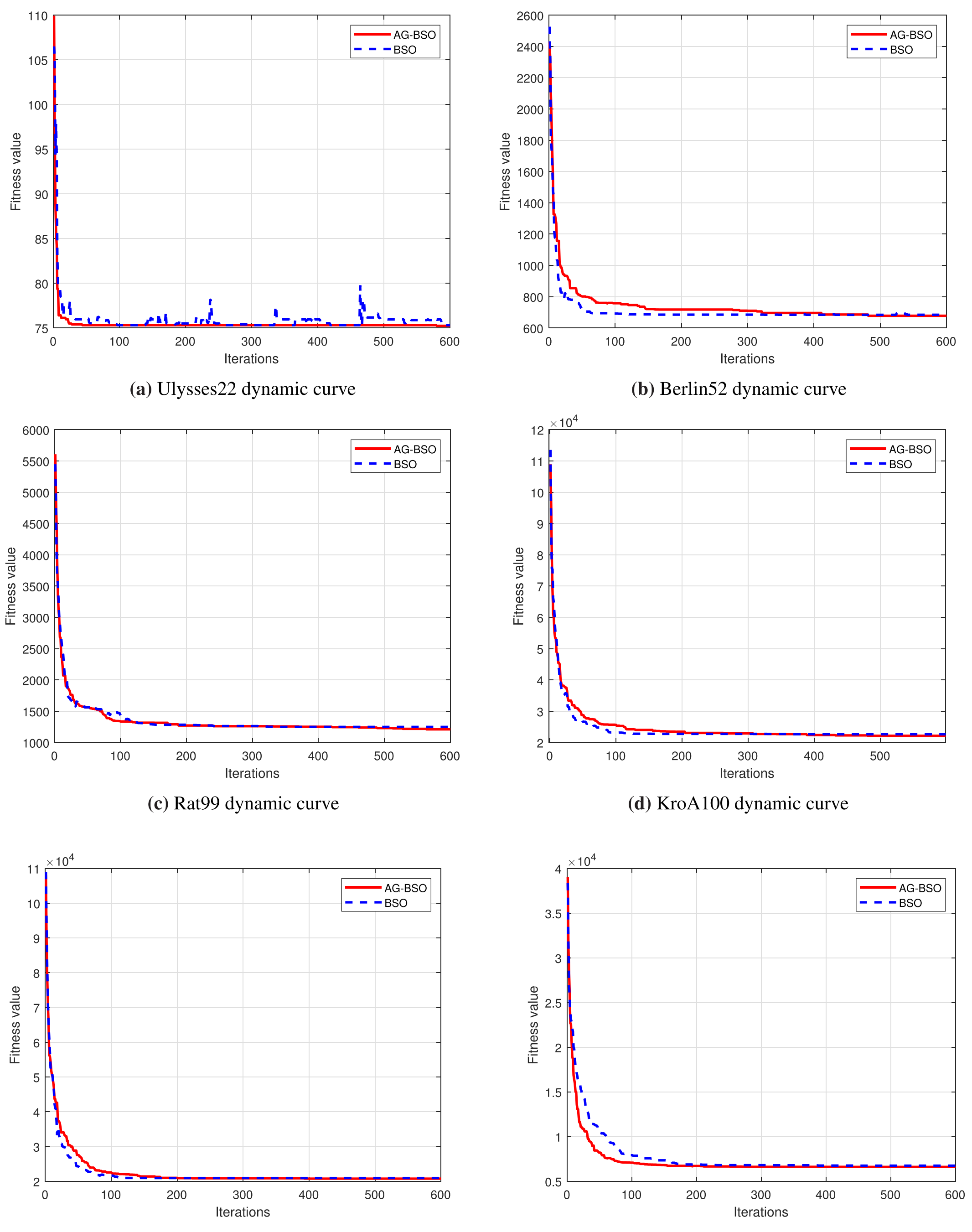
Note: DEV indicates that the standard deviation.

To test the performance of the components of the AG-BSO algorithm and observe their impact relative to the original BSO algorithm, each of the three components was added to the BSO algorithm individually for experimental comparisons with the original BSO algorithm on the data sets ulyssses22, eil51, berlin52, ch150, and pr226. The results are shown in TABLE.2. The detailed definitions of the algorithm variants with the different components are given as follows:

1) The improved BSO algorithm obtained by implementing only hierarchical clustering is denoted by BSO1;
2) The improved BSO algorithm obtained by implementing only the exchange rule and the heuristic crossover operator for updating individuals is denoted by BSO2;
3) The improved BSO algorithm obtained by implementing only the greedy algorithm to initialize the population is denoted by BSO3.

At the same time.we experimentally analyzed the characteristics of the new components, and the results are presented below:

1) The introduction of hierarchical clustering greatly reduces the solution time of the original BSO algorithm, but the algorithm can easily fall into a local optimum, and the solution accuracy cannot be guaranteed. When BSO1 is used to solve the TSP instances ulysses22, eil51 and berlin52, the solution times(in seconds) are 13.05, 20.43, and 10.93, respectively. However, the solution accuracy is poor because of the susceptibility to local optima; the optimized values are lower than those achieved with the original BSO algorithm. When BSO1 is used to solve the TSP instance pr226, the time spent is 120.13, which is 67.2 seconds faster than the original BSO algorithm but longer than the solution time of BSO3;
2) The incorporation of the exchange rule and the heuristic crossover operator into the BSO algorithm can help the algorithm jump out of local optima and enhance its global optimization capability. However, this comes at the cost of an increase in the solution time. When BSO2 is used to solve the TSP instances ulysses22, eil51, berlin52 and pr226, the optimized values are better than those obtained using the original BSO algorithm, but the solution time is significantly increased. When BSO2 is used to solve pr226, the solution time is 211.47, which is 24.14 longer than that of the original BSO algorithm;



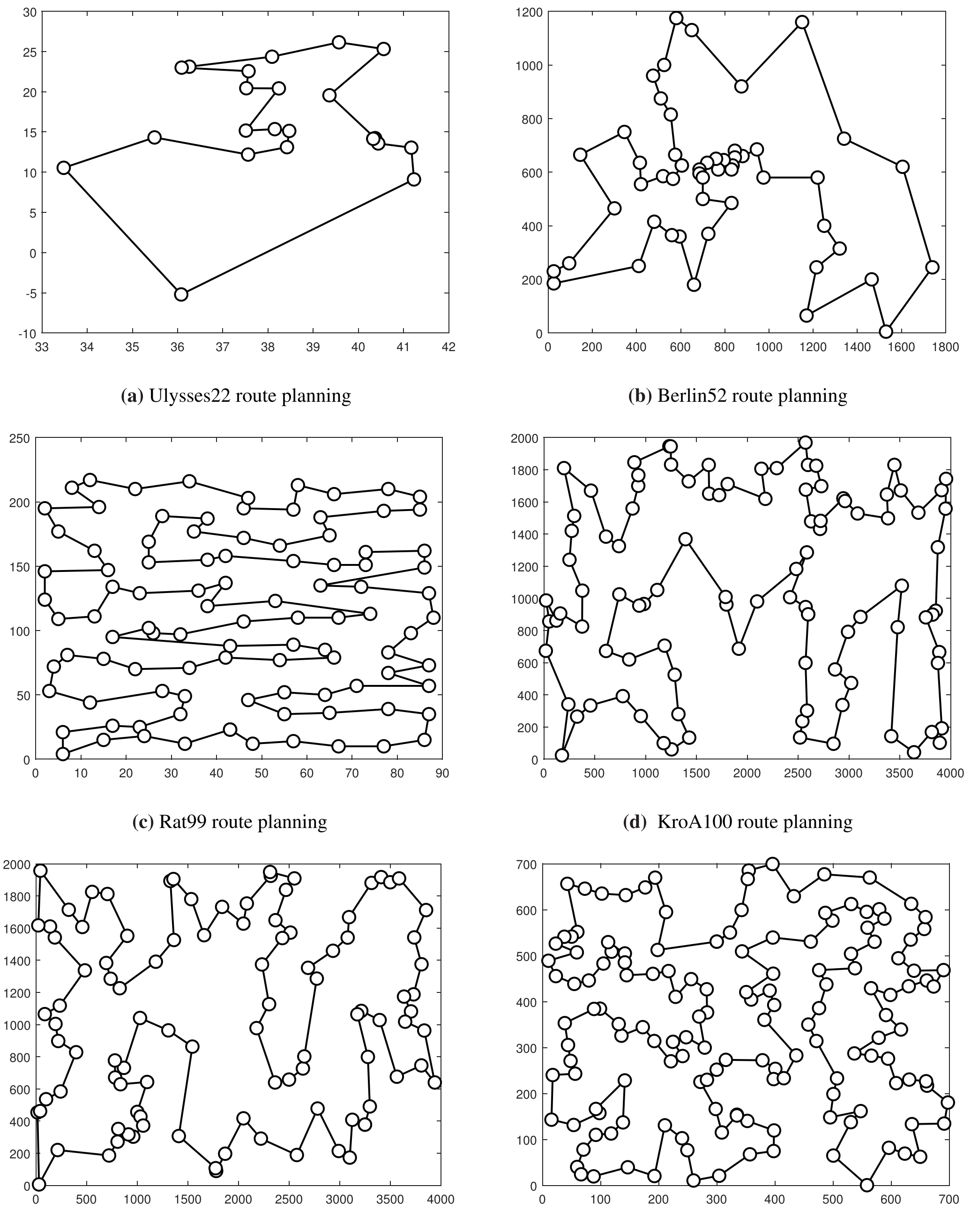
(a) Ulysses22 dynamic curve **(b)** Berlin52 dynamic curve **(c)** Rat99 dynamic curve **(d)** KroA100 dynamic curve (e) KroC100 dynamic curve **(f)** Ch150 dynamic curve

FIGURE 4. Dynamic average convergence curves of AG-BSO and the basic BSO algorithm.

3) The use of the greedy algorithm to initialize the population for a large-scale TSP instance can effectively reduce the solution time of the original BSO algorithm. When BSO3 is used to solve the TSP instances ulysses22, eil51 and berlin52, the solution time and solution accuracy show no major changes compared with the original BSO algorithm. However, the time taken to solve the large-scale instance pr226 is 87.12, which is 100.21 faster than the original BSO algorithm.

B. COMPARISON OF THE ORIGINAL AND IMPROVED BSO

As mentioned in the introduction to this section, an experiment was conducted to prove that the improved BSO performs better than the basic version of the original BSO. In this experiment, 28 TSP instances were included. The experimental results are shown in TABLE.3. The best solution found by the algorithm, the standard deviation (calculated as shwon in Eq.(7)), and the average runn time (in seconds) are shown for



(a) Ulysses22 route planning

(b) Berlin52 route planning

(c) Rat99 route planning

(d) KroA100 route planning

(e) KroC100 route planning

(f) Ch150 route planning

FIGURE 5. Some examples of optimal paths obtained by the improved BSO.

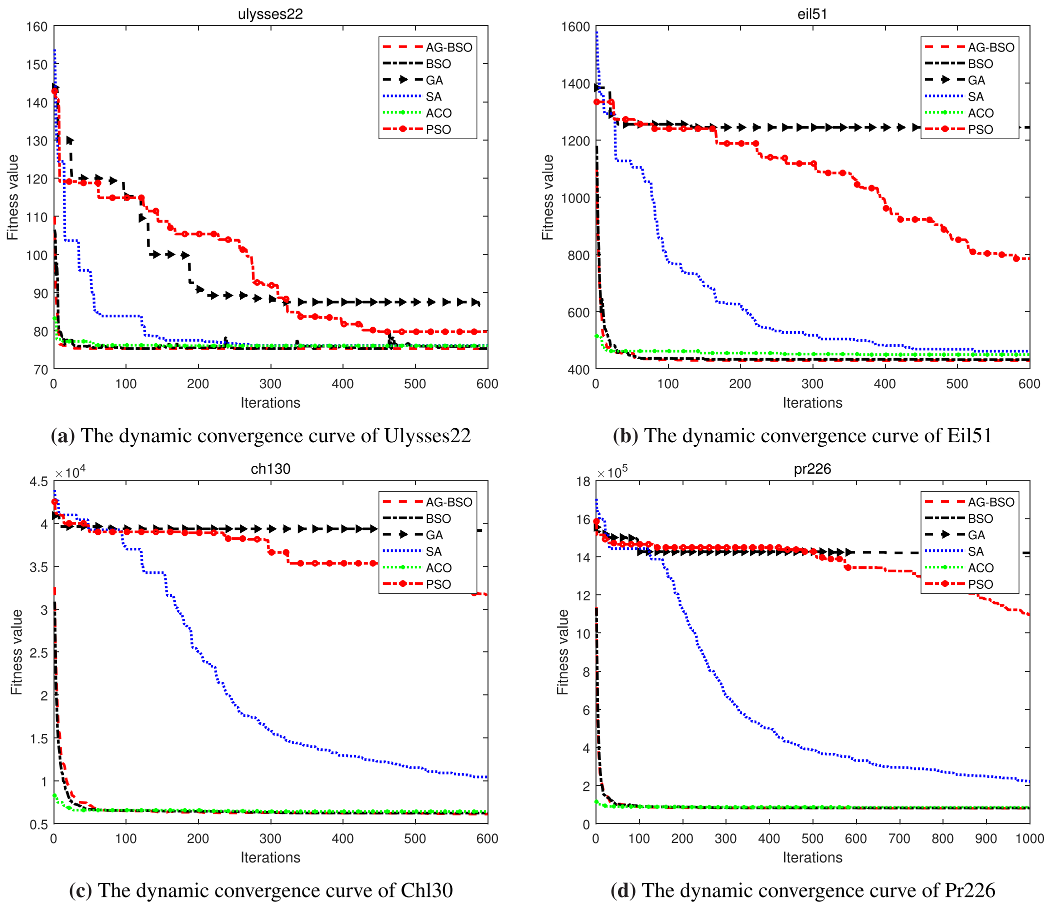
each instance.

$$Dev = \frac{OV - IV}{IV} \tag{7}$$

where Dev represents the deviation rate, OV represents the best value found, and IV represents the known ideal value.

Based on the problem size, the TSP instances selected for this experiment can be divided into two categories. When the number of city nodes in a problem instance is fewer than or equal to 150, the instance belongs to category 1; otherwise, it belongs to category 2 and is considered a large-scale TSP instance. When the number of city nodes is fewer than or equal to 150, the maximum number of iterations is 600; when the number of city nodes is greater than 150, the maximum number of iterations is 1000. An algorithm terminates when the preset maximum number of iterations is reached.

From the results shown in TABLE.3, it can be concluded that for these 28 examples, the improved BSO algorithm is superior to the basic BSO algorithm in terms of the best value, runn time and standard deviation. It can be clearly seen from TABLE.3 that to solve the TSP with more than 100 city



(a) The dynamic convergence curve of Ulysses22

(b) The dynamic convergence curve of Eil51

(c) The dynamic convergence curve of Chl30

(d) The dynamic convergence curve of Pr226

FIGURE 6. **The dynamic average convergence curves.**

nodes, the improved algorithm takes significantly less time than the basic BSO algorithm and the accuracy of the solution is higher than that of the basic BSO algorithm.Furthermore, AG-BSO achieves the known optimal values for ulysses22, berlin52, rat99, kroA100, kroC100 and ch150. Notably, the best values found for data sets ulysses22 and kroA100 are lower than the known optimal values, which is normal. Most of the data sets used in the existing literature on TSP problems are derived from the TSBLIB test set released by Reinelt in 1991 [67]. From a review of a large number of previous studies, it can be found that the values obtained by many algorithms in these studies are also lower than the known optimal values. For example, in [74], a multiobjective ant colony algorithm was used to solve ulysses22, and the optimal value found was 75.31; in [72], a metaheuristic hybrid algorithm was used to solve ulysses22, att488, and berlin52, and the obtained values were 56.52, 13908.4, and 5970.83, respectively, all lower than the known optimal values. respectively. In the 20 experimental results, the standard deviation was less than 1%, accounting for 71% of all instances. The performance comparison between the improved BSO and the basic BSO is shown in Fig.4. Fig.5 shows the experimental results of some examples of the known optimal solutions.

C. EXPERIMENTATION WITH THE IMPROVED BSO AND TRADITIONAL INTELLIGENT ALGORITHMS

To prove that the improved algorithm is better than the traditional intelligent algorithms, the four specific instances ulysses22, eil51, ch130 and pr226 were selected for the simulation experiments presented in this section. Each intelligent algorithm was independently run 30 times on each TSP instance. To ensure the fairness of the comparison and the obtain statistical results quickly, for the same TSP instances, all algorithms are tested on the identical operating system on the same computer. Fig.6 shows the optimization processes of the six tested algorithms for solving TSP instances ulysses22, eil51, ch130, and pr226.

From the information presented in Fig.6, it can be intuitively seen that for testing TSP data ulysses22, eil51, and ch130, the fixed number of iterations was set to 600. For these problem instances, the AG-BSO algorithm proposed in this article converges to the optimal value in the 44th, 184th and 536th generations, respectively, while the optimal values obtained by the SA, GA, and PSO algorithms continue to constantly change with an increasing number of iterations. When the maximum number of iterations is reached, these algorithms have not yet converged; thus, it can be seen that the optimization efficiency of these three algorithms

TABLE 4. Experimental results of the BSO algorithm and other traditional algorithms.

Algorithm	Index	ulysses22	eil51	ch130	pr226
AG-BSO	MeanV	**75.24**	**428.58**	**6115.25**	**80961.17**
	MeanT	9.05	12.19	**26.69**	**66.63**
	Dev	**0**	**0.61**	**0.07**	**0.74**
BSO	MeanV	75.31	431.91	6233.18	82174.13
	MeanT	15.05	24.97	67.11	187.33
	Dev	0	1.39	2	2.25
GA	MeanV	86.61	1244.29	39160.18	141984.76
	MeanT	2.38	5.37	30.07	178.05
	Dev	14.46	192.09	540.83	76.67
SA	MeanV	76.09	461.81	10363.34	221305.62
	MeanT	35.87	41.03	36.16	91.64
	Dev	0.56	8.64	69.59	175.36
ACO	MeanV	76.06	449.72	6419.74	86529.16
	MeanT	12.13	36.9	156.2	191.87
	Dev	0.52	5.57	5.05	7.66
PSO	MeanV	79.73	785.25	31516.05	109647.38
	MeanT	10.32	6.14	28.05	106.93
	Dev	5.37	84.33	1315.99	36.43

TABLE 5. Parameter settings for three improved BSO algorithms.

Algorithm	Year	Parameters
AG-BSO	2020	p_one=0.6, p_two=0.4, p_one_center=0.45, p_two_center=0.5
MDBSO	2019	$P6b$=0.5, $P6biii$=0.5, $P6c4$=0.7
PKDBSO	2018	$P6b$=0.5, $P6biii$=0.5, $P6c$=0.7

is worse than that of the AG-BSO. For the TSP instance pr226, the number of iterations was increased to 1000. The optimization capabilities of GA, PSO, and SA are still far from those of the AG-BSO algorithm. The convergence speed of ACO is faster than that of other algorithms, however, there is a large difference between the optimal value obtained by ACO and that obtained by AG-BSO.

The k-means clustering method is sensitive to noise and outliers. This causes the value found by that the original BSO to oscillate around the optimal value in the later stages of algorithm execution. The AG-BSO algorithm avoids this problem by using agglomerative hierarchical clustering. To further illustrate the advantages of AG-BSO in terms of optimization efficiency and accuracy, the optimal values found by each algorithm and the specific amounts of time spent are presented in detail TABLE.4. In this table, *Dev* is the standard deviation. *MeanV* represents the average of all the best results in the 30 individual runs. *MeanT* represents the average corresponding average run time.

Taking the eii51 data set as an example, it can be seen that the deviation rate of the AG-BSO algorithm is only 0.61%, representing a great advantage compared with the other algorithms; in particular, this deviation rate is reduced by 83.72% compared with that of the PSO algorithm. TABLE.4 shows that the AG-BSO algorithm also has obvious advantages

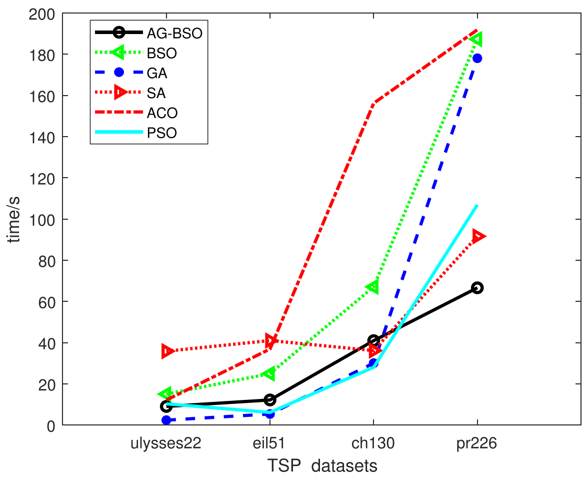


FIGURE 7. Run time visualization.

compared with the other algorithms in terms of the optimization speed, deviation rate and stability when solving problems involving small volumes of TSP data. Compared with the original BSO algorithm, AG-BSO is superior in terms of both solution time and solution accuracy, demonstrating that the improved algorithm has better optimization ability and greater robustness. Fig.7 illustrates the execution times of the six algorithms for solving these four TSP instances.

D. COMPARISON OF AG-BSO WITH OTHER IMPROVED ALGORITHMS

On the basis of the above comparisons the AG-BSO algorithm with traditional algorithms, this section compares AG-BSO with other improved algorithms.

First, AG-BSO is compared with two other improved BSO algorithms in the literature that participated in the comparison are as follows: (1) the improved BSO in reference [48]

TABLE 6. Experimental results.

Instance		AG-BSO		MDBSO		PKDBSO	
Name	Optima	Best	Dev(%)	Best	Dev(%)	Best	Dev(%)
eil51	426.00	428.58	**0.61**	436.40	2.44	452.80	6.29
st70	675.00	678.00	**0.44**	685.50	1.51	694.30	2.80
kroA100	21282.00	21178.00	**0.00**	21339.00	0.26	22023.00	3.41
ch150	6528.00	6528.00	**0.00**	6939.00	6.29	7381.00	13.06
pr226	80369.00	80961.17	**0.74**	81606.00	1.53	85526.00	6.41
pcb442	50778.00	52375.32	**3.15**	52467.00	3.32	58516.00	15.23

TABLE 7. Parameter settings for the five improved algorithms.

Algorithm	Year	Parameters
AG-BSO(this paper)	2020	p_one=0.6, p_two=0.4, p_one_center=0.45, p_two_center=0.5
GACO	2020	α=1, β=2, ρ=0.9, N=30,$bestN$=30, $MaxIt$=1000, $mutations$=1000
IACO	2019	α=1, β=2, ρ=0.05, Q=30, $NCmax$=1000
DWCA	2018	Psr=10, $Pstr$=38, R=5%
OMACO	2014	α=1, β=2, ρ=0.05, Q=120,

TABLE 8. Statistical results of AG-BSO and four other algorithms for solving TSP instances.

Instance		AG-BSO		GACO		OMACO		IACO		DWCA	
Name	Optima	Best	Dev(%)	Best	Dev(%)	Best	Dev(%)	Best	Dev(%)	Best	Dev(%)
ulysses22	75.67	75.24	0.00	NA	NA	75.31	0	NA	NA	NA	NA
att48	33523.71	33538.34	0.05	35298	5.29	34379.70	2.55	33522	0.00	NA	NA
eil51	426.00	428.58	0.61	NA	NA	428.87	0.67	426	0.00	426	0.00
berlin52	7542.00	7542.00	0.00	7835.30	3.89	7562.39	0.27	7542.00	0.00	7542.00	0.00
st70	675.00	678.00	0.44	NA	NA	678.62	0.54	676.00	0.15	678.60	0.53
eil76	538.00	540.69	0.50	NA	NA	556.94	3.52	538.00	0.00	543.00	0.93
kroA100	21282.00	21070.09	0.00	NA	NA	21320.96	0.18	21308	0.12	21282.00	0.00
kroB100	22140.00	22152.00	0.05	NA	NA	22196.53	0.26	NA	NA	22178.00	0.17
kroC100	20749.00	20749.00	0.00	NA	NA	21294.39	2.63	NA	NA	21529.00	3.76
eil101	629.00	633.00	0.64	NA	NA	642.03	2.07	631.00	0.32	639.00	1.59
pr226	80369.00	80961.17	0.74	NA	NA	81382.45	1.26	NA	NA	NA	NA
pcb442	50778.00	52375.32	3.15	NA	NA	53842.84	6.04	NA	NA	NA	NA
pr439	107217.00	113074.50	5.46	NA	NA	113797.02	6.14	NA	NA	NA	NA

Note: NA means that the algorithm is not running on the relevant TSP data set.

is abbreviated as PKDBSO; (2) the proposed reinforced brain storm optimization in reference [70] is abbreviated as MDBSO; among the three improved BSO algorithms, the maximum number of iterations is 1000, and the population number is 100. The detailed parameter settings of different algorithms are shown in TABLE.5.

TABLE.6 shows the simulation results of the different algorithms at various scales. When tested kroA100, AG-BSO finds a best value of 21178.00, better than the ideal value. Obviously, the result for AG-BSO solving 51 cities is 0.69%, which is 1.75% lower than the result for MDBSO(2.44%). Similarly, for st70, AG-BSO has a deviation rate of 0.44%, which is lower than the previous best deviation of 1.5%. For kroA100, ch150, pr226, and pcb442, the deviation rate of AG-BSO are reduced by 0.74%, 4.88%, 0.8%, and 0.78%, respectively, compared with those of MDBSO. In summary, the experimental data show that the proposed method achieves good results. As the problem complexity increases, AG-BSO can jump out of local optimal solution faster, improving the convergence performance and speed. The reason for this is that hierarchical clustering is used to obtain solutions close to the global optimum, thereby improving the convergence properties in the population initialization stage. The experimental results show that, compared with the MDBSO algorithm, the AG-BSO algorithm can better avoid falling into local optimal when solving TSP instances, and thus can find better results more efficiently and stably. It can be concluded that AG-BSO is significantly superior to MDBSO and PKDBSO.

Finally, AG-BSO was compared with four improved algorithms in the literature. The four improved algorithms that participated in the comparison are as follows:

(1) the proposed meta-heuristic hybrid algorithm in reference [72] is abbreviated as GACO; (2) the improved ACO is abbreviated as IACO in literature [71]; (3) the proposed discrete water cycle algorithm is abbreviated as DWCO in literature [73]; (4) the proposed multi-role ACO is OMACO in reference [74]. The detailed parameter settings of the different algorithms are shown in TABLE.7.

TABLE.8 shows the statistical results of AG-BSO and the other improved algorithms. The conclusions obtained from Table.8 are similar to the previous conclusions. Once again, AG-BSO is the best-performing technique overall. Among the small-scale TSP instances, better results are obtained for ulysses22, att48, berlin52, st70, eil76, kroA100, kroC100 and eil101. Additionally, for the large-scale TSP instacnes pr226, pcb442 and pr439, the AG-BSO proposed in this article performs significantly better than the other improved algorithms. In addition, as the problem scale increases, the computational complexity also increases significantly, and the performance of the other algorithms drops sharply; by contrast, AG-BSO maintains good adaptability for all TSP instances.

V. CONCLUSION

The improved BSO proposed in the literature tend to have a slow convergence speed, can easily fall into local optima and have a low solution accuracy when used to solve complex combinatorial problems. In addition, although most of the current algorithms reported in the literature can achieve close to ideal results when solving small-scale TSP instacnes, they show poor performance for large-scale instances. To address the above problems, this article proposes an improved BSO algorithm named AG-BSO. The three core improvements of AG-BSO are as follows: (1) the k-means clustering algorithm used in the original BSO algorithm is replaced with agglomerative hierarchical clustering; (2) the greedy algorithm is introduced in AG-BSO; and (3) an exchange rules and a heuristic crossover operators are adopted for updating individuals in the population.

On the basis of in-depth theoretical research on the standard BSO algorithm and a deep analysis of the TSP, to address the lack of robustness of the standard BSO algorithm and other issues, an agglomerative greedy BSO algorithm is proposed for the first time in this work for solving the TSP. To prove that the proposed AG-BSO is a promising approximate method for solving the TSP, we have compared its performance with that of the basic BSO algorithm on 28 TSP instances. In addition, the results of the proposed algorithm have been compared with four different traditional metaheuristic algorithms: GA, SA, ACO and PSO, and with the results of six improved intelligent algorithms. Overall, the simulation results show that AG-BSO is superior in terms of optimization efficiency, convergence speed and robustness. The AG-BSO demonstrates excellent performance for solving the considered TSP instances, better than the performance of the other improved algorithms in most cases.

The TSP is a classic combinatorial optimization problem. However, the conclusions obtained based on the TSP for the improved algorithm proposed in this article cannot be generalized to other combinatorial optimization problems. Therefore, in future work, we plan to develop an improved BSO to solve other routing problems such as vehicle routing problems. Moreover, there are a large number of meta-heuristic algorithms available in the literature for solving the TSP. Here, the AG-BSO algorithm has been compared with six selected improved algorithms, and the simulation results showed that the proposed AG-BSO is promising. Nevertheless, we believe that conducting more extensive experiments on additional technologies is valuable to the scientific community. The efficiency of the AG-BSO algorithm in solving large-scale TSP instances is exciting, but in most cases it cannot reach the known optimal value. Further improving the algorithm will be a major challenge to address in subsequent work. Future strategies for improving the performance of BSO will focus on the mechanism for updating the individuals in the population, the diversity of the population, and the incorporation of other novel intelligent optimization algorithms.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers and Academic Editor for their valuable and constructive comments, which greatly improved the quality and integrity of this manuscript.

REFERENCES

[1] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Human Sci.*, Oct. 1995, pp. 39–43.

[2] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

[3] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.

[4] I. Fister, I. Fister, X.-S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm Evol. Comput.*, vol. 13, pp. 34–46, Dec. 2013.

[5] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, May 2016.

[6] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proc. World Congr. Nature Biologically Inspired Comput. (NaBIC)*, Dec. 2009, pp. 210–214.

[7] J. Xue and B. Shen, "A novel swarm intelligence optimization approach: Sparrow search algorithm," *Syst. Sci. Control Eng.*, vol. 8, no. 1, pp. 22–34, Jan. 2020.

[8] Y. Shi, "Brain storm optimization algorithm," in *Proc. Int. Conf. Swarm Intell.*, 2011, pp. 303–309.

[9] M. Arsuaga-Ríos and M. A. Vega-Rodríguez, "Multi-objective energy optimization in grid systems from a brain storming strategy," *Soft Comput.*, vol. 19, no. 11, pp. 3159–3172, Nov. 2015.

[10] H. B. Duan, S. T. Li, and Y. H. Shi, "Predator–Prey brain storm optimization for DC brushless motor," *IEEE Trans. Magn.*, vol. 49, no. 10, pp. 5336–5340, Oct. 2013.

[11] H. Qiu and H. Duan, "Receding horizon control for multiple UAV formation flight based on modified brain storm optimization," *Nonlinear Dyn.*, vol. 78, no. 3, pp. 1973–1988, Nov. 2014.

[12] G. Xiong, D. Shi, J. Zhang, and Y. Zhang, "A binary coded brain storm optimization for fault section diagnosis of power systems," *Electr. Power Syst. Res.*, vol. 163, pp. 441–451, Oct. 2018.

[13] Y. Wang, S. Gao, Y. Yu, and Z. Xu, "The discovery of population interaction with a power law distribution in brain storm optimization," *Memetic Comput.*, vol. 11, no. 1, pp. 65–87, Mar. 2019.

[14] Z. J. Cao, "An improved brain storm optimization with differential evolution strategy for applications of ANNs," *Math. Problems Eng.*, vol. 2015, Sep. 2015, Art. no. 923698.

[15] J. Wang, R. Hou, C. Wang, and L. Shen, "Improved v-support vector regression model based on variable selection and brain storm optimization for stock price forecasting," *Appl. Soft Comput.*, vol. 49, pp. 164–178, Dec. 2016.

[16] X. Ma, Y. Jin, and Q. Dong, "A generalized dynamic fuzzy neural network based on singular spectrum analysis optimized by brain storm optimization for short-term wind speed forecasting," *Appl. Soft Comput.*, vol. 54, pp. 296–312, May 2017.

[17] H. Liang, Z. Wang, and Y. Liu, "A new hybrid ant colony optimization based on brain storm optimization for feature selection," *IEICE Trans. Inf. Syst.*, vol. 102, no. 7, pp. 1396–1399, 2019.

[18] D. Palanikkumar and S. Priya, "Brain storm optimization graph theory (BSOGT) and energy resource aware virtual network mapping (ERVNM) for medical image system in cloud," *J. Med. Syst.*, vol. 43, no. 2, p. 10, Feb. 2019.

[19] Y. Wu, X. Wang, Y. Fu, and G. Li, "Many-objective brain storm optimization algorithm," *IEEE Access*, vol. 7, pp. 186572–186586, 2019.

[20] Y. Shen, M. Liu, J. Yang, Y. Shi, and M. Middendorf, "A hybrid swarm intelligence algorithm for vehicle routing problem with time windows," *IEEE Access*, vol. 8, pp. 93882–93893, 2020.

[21] A. Cervantes-Castillo and E. Mezura-Montes, "A modified brain storm optimization algorithm with a special operator to solve constrained optimization problems," *Appl. Intell.*, pp. 1–17, Jul. 2020, doi: 10.1007/s10489-020-01763-8.

[22] A. Bhatt, P. Dimri, and A. Aggarwal, "Self-adaptive brainstorming for jobshop scheduling in multicloud environment," *Softw., Pract. Exper.*, vol. 50, no. 8, pp. 1381–1398, 2020.

[23] Z. Xu, X. Li, X. Meng, and Y. Liu, "A distributed brain storm optimization for numerical optimization and graph planarization," *IEEE Access*, vol. 7, pp. 39770–39781, 2019.

[24] R. A. Ibrahim, "Galaxy images classification using hybrid brain storm optimization with moth flame optimization," *J. Astronomical Telescopes Instrum. Syst.*, vol. 4, no. 3, 2018, Art. no. 038001.

[25] C. Narmatha, S. M. Eljack, A. A. R. M. Tuka, S. Manimurugan, and M. Mustafa, "A hybrid fuzzy brain-storm optimization algorithm for the classification of brain tumor MRI images," *J. Ambient Intell. Humanized Comput.*, pp. 1–9, Aug. 2020.

[26] Y. Sun, "A hybrid approach by integrating brain storm optimization algorithm with grey neural network for stock index forecasting," *Abstract Appl. Anal.*, vol. 2014, May 2014, Art. no. 759862.

[27] K. B. Boamah, J. Du, D. Adu, C. N. Mensah, L. Dauda, and M. A. S. Khan, "Predicting the carbon dioxide emission of China using a novel augmented hypo-variance brain storm optimisation and the impulse response function," *Environ. Technol.*, pp. 1–13, May 2020.

[28] X. Xue and J. Lu, "A compact brain storm algorithm for matching ontologies," *IEEE Access*, vol. 8, pp. 43898–43907, 2020.

[29] J. Chen, "Enhanced brain storm optimization algorithm for wireless sensor networks deployment," in *Proc. Int. Conf. Swarm Intell.*, 2015, pp. 373–381.

[30] Y. Shi, "Brain storm optimization algorithm in objective space," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, May 2015, pp. 1227–1234.

[31] S. Dash, D. Joshi, and G. Trivedi, "Multiobjective analog/RF circuit sizing using an improved brain storm optimization algorithm," *Memetic Comput.*, vol. 10, no. 4, pp. 423–440, Dec. 2018.

[32] H. Duan, D. Zhang, Y. Shi, and Y. Deng, "Close formation flight of swarm unmanned aerial vehicles via metric-distance brain storm optimization," *Memetic Comput.*, vol. 10, no. 4, pp. 369–381, Dec. 2018.

[33] C. Li, D. Hu, Z. Song, F. Yang, Z. Luo, J. Fan, and P. X. Liu, "A vector grouping learning brain storm optimization algorithm for global optimization problems," *IEEE Access*, vol. 6, pp. 78193–78213, 2018.

[34] Z. Song, J. Peng, C. Li, and P. X. Liu, "A simple brain storm optimization algorithm with a periodic quantum learning strategy," *IEEE Access*, vol. 6, pp. 19968–19983, 2018.

[35] Z. Cao and L. Wang, "An active learning brain storm optimization algorithm with a dynamically changing cluster cycle for global optimization," *Cluster Comput.*, vol. 22, no. 4, pp. 1413–1429, Dec. 2019.

[36] W. Chen, Y. Cao, S. Cheng, Y. Sun, Q. Liu, and Y. Li, "Simplex search-based brain storm optimization," *IEEE Access*, vol. 6, pp. 75997–76006, 2018.

[37] Y. H. Sun, "Brain storm optimization using a slight relaxation selection and multi-population based creating ideas ensemble," *Appl. Intell.*, vol. 50, pp. 3137–3161, May 2020.

[38] S. Peng, H. Wang, and Q. Yu, "Multi-clusters adaptive brain storm optimization algorithm for QoS-aware service composition," *IEEE Access*, vol. 8, pp. 48822–48835, 2020.

[39] H. B. Duan and C. Li, "Quantum-behaved brain storm optimization approach to solving Loney's solenoid problem," *IEEE Trans. Magn.*, vol. 51, no. 1, 2015, Art. no. 7000307.

[40] M. El-Abd, "Global-best brain storm optimization algorithm," *Swarm Evol. Comput.*, vol. 37, pp. 27–44, Dec. 2017.

[41] C. Li, Z. Song, J. Fan, Q. Cheng, and P. X. Liu, "A brain storm optimization with multi-information interactions for global optimization problems," *IEEE Access*, vol. 6, pp. 19304–19323, 2018.

[42] J. P. Papa, G. H. Rosa, A. N. de Souza, and L. C. S. Afonso, "Feature selection through binary brain storm optimization," *Comput. Electr. Eng.*, vol. 72, pp. 468–481, Nov. 2018.

[43] Y. Guo, H. Yang, M. Chen, D. Gong, and S. Cheng, "Grid-based dynamic robust multi-objective brain storm optimization algorithm," *Soft Comput.*, vol. 24, no. 10, pp. 7395–7415, May 2020.

[44] Y. Yu, S. Gao, Y. Wang, Z. Lei, J. Cheng, and Y. Todo, "A multiple diversity-driven brain storm optimization algorithm with adaptive parameters," *IEEE Access*, vol. 7, pp. 126871–126888, 2019.

[45] L. Ke, "A brain storm optimization approach for the cumulative capacitated vehicle routing problem," *Memetic Comput.*, vol. 10, no. 4, pp. 411–421, Dec. 2018.

[46] X. R. Chen, "An improved brain storm optimization for a hybrid renewable energy system," *IEEE Access*, vol. 7, pp. 46513–46526, 2019.

[47] J.-H. Hao, J.-Q. Li, Y. Du, M.-X. Song, P. Duan, and Y.-Y. Zhang, "Solving distributed hybrid flowshop scheduling problems by a hybrid brain storm optimization algorithm," *IEEE Access*, vol. 7, pp. 66879–66894, 2019.

[48] Y. Xu, Y. Wu, Y. Fu, X. Wang, and A. Lu, "Discrete brain storm optimization algorithm based on prior knowledge for traveling salesman problems," in *Proc. 13th IEEE Conf. Ind. Electron. Appl. (ICIEA)*, May 2018, pp. 2740–2745.

[49] S. Thanga Revathi, N. Ramaraj, and S. Chithra, "Brain storm-based whale optimization algorithm for privacy-protected data publishing in cloud computing," *Cluster Comput.*, vol. 22, no. S2, pp. 3521–3530, Mar. 2019.

[50] O. Osicka, M. Guajardo, and K. Jörnsten, "Cooperation of customers in traveling salesman problems with profits," *Optim. Lett.*, vol. 14, no. 5, pp. 1219–1233, Jul. 2020.

[51] W. Deng, J. Xu, and H. Zhao, "An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem," *IEEE Access*, vol. 7, pp. 20281–20292, 2019.

[52] S. S. Choong, L.-P. Wong, and C. P. Lim, "An artificial bee colony algorithm with a modified choice function for the traveling salesman problem," *Swarm Evol. Comput.*, vol. 44, pp. 622–635, Feb. 2019.

[53] S. Cheng, Q. Qin, J. Chen, and Y. Shi, "Brain storm optimization algorithm: A review," *Artif. Intell. Rev.*, vol. 46, no. 4, pp. 445–458, Dec. 2016.

[54] M. J. Santos, P. Amorim, A. Marques, A. Carvalho, and A. Póvoa, "The vehicle routing problem with backhauls towards a sustainability perspective: A review," *TOP*, vol. 28, no. 2, pp. 358–401, Jul. 2020.

[55] M. Drexl and M. Schneider, "A survey of variants and extensions of the location-routing problem," *Eur. J. Oper. Res.*, vol. 241, no. 2, pp. 283–308, Mar. 2015.

[56] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *Eur. J. Oper. Res.*, vol. 59, no. 3, pp. 345–358, Jun. 1992.

[57] Q.-K. Pan and R. Ruiz, "An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem," *Omega*, vol. 44, pp. 41–50, Apr. 2014.

[58] S. Meeran and A. Share, "Optimum path planning using convex hull and local search heuristic algorithms," *Mechatronics*, vol. 7, no. 8, pp. 737–756, Dec. 1997.

[59] Z.-H. Zhan, J. Zhang, Y.-H. Shi, and H.-l. Liu, "A modified brain storm optimization," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2012, pp. 1–8.

[60] L.-H. Chen, S.-Y. Hsieh, L.-J. Hung, and R. Klasing, "Approximation algorithms for the p-hub center routing problem in parameterized metric graphs," *Theor. Comput. Sci.*, vol. 806, pp. 271–280, Feb. 2020.

[61] G. Pan, K. Li, A. Ouyang, and K. Li, "Hybrid immune algorithm based on greedy algorithm and delete-cross operator for solving TSP," *Soft Comput.*, vol. 20, no. 2, pp. 555–566, Feb. 2016.

[62] L. He, B. Agard, and M. Trépanier, "A classification of public transit users with smart card data based on time series distance metrics and a hierarchical clustering method," *Transportmetrica A: Transp. Sci.*, vol. 16, no. 1, pp. 56–75, Dec. 2020.

[63] J.-F. Huang, S.-S. Kao, S.-Y. Hsieh, and R. Klasing, "Top-down construction of independent spanning trees in alternating group networks," *IEEE Access*, vol. 8, pp. 112333–112347, 2020.

[64] S.-Y. Hsieh, R. Klasing, C. Luo, and S.-C. Tsai, "Special issue on pervasive systems, algorithms, and networks foreword," *J. Inf. Sci. Eng.*, vol. 35, no. 4, 2019.

[65] S.-Y. Hsieh, S.-S. Kao, and Y.-S. Lin, "A swap-based heuristic algorithm for the maximum k-Plex problem," *IEEE Access*, vol. 7, pp. 110267–110278, 2019.

[66] K. Li, F. Xu, P. Huang, and W. Zhang, "A new best-worst ant system with heuristic crossover operator for solving TSP," in *Proc. 5th Int. Conf. Natural Comput.*, vol. 4, Aug. 2009, pp. 92–97.

[67] G. Reinelt, "TSPLIB—A traveling salesman problem library," *Informs J. Comput.*, vol. 3, no. 4, pp. 376–384, 1991.

[68] K. De Jong, "Adaptive system design: A genetic approach," *IEEE Trans. Syst., Man, Cybern.*, vol. 10, no. 9, pp. 566–574, Sep. 1980.

[69] A. E.-S. Ezugwu, A. O. Adewumi, and M. E. Frîncu, "Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem," *Expert Syst. Appl.*, vol. 77, pp. 189–210, Jul. 2017.

[70] Y. Wu, "An adaptive brain storm optimization algorithm based on heuristic operators for TSP," in *Proc. Int. Conf. Bio-Inspired Comput., Theories Appl.*, 2019, pp. 662–672.

[71] D. Qiao, "Research on improving ant colony algorithm to solve TSP problem," *Machinery Des. Manuf.*, no. 10, pp. 144–149, 2019.

[72] U. Ashraf, "Meta-heuristic hybrid algorithmic approach for solving combinatorial optimization problem (TSP)," in *Proc. Int. Conf. Bio-Inspired Comput., Theories Appl.*, 2019, pp. 622–633.

[73] E. Osaba, J. D. Ser, A. Sadollah, M. N. Bilbao, and D. Camacho, "A discrete water cycle algorithm for solving the symmetric and asymmetric traveling salesman problem," *Appl. Soft Comput.*, vol. 71, pp. 277–290, Oct. 2018.

[74] D. Pengzhen, T. Zhenmin, and S. Yan, "An object-oriented multi-role ant colony optimization algorithm for solving TSP problem," *Control Decis.*, vol. 29, no. 10, pp. 1729–1736, 2014.

CHANGYOU WU received the B.E. and M.E. degrees in industrial engineering and management science engineering from Northeast Agricultural University, China, in 2004 and 2006, respectively, and the Ph.D. degree in mechanized engineering from Northeast Agricultural University, China, in 2009.

He is currently an Associate Professor with the Engineering College, Shandong Institute of Business and Technology. He has published over 30 articles in domestic and international academic journals and conference proceedings. His current research interests include genetic algorithm theory and its application, neural network theory and its application, parallel computing, and water resources management.

XISONG FU received the B.E. degree in information management and information system from the Institute of Technology, East China Jiaotong University, China, in 2019. He is currently pursuing the master's degree with the Engineering College, Shandong Institute of Business and Technology.

His current research interest includes brain storm optimization algorithm theory and its application.