Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 9461861, 15 pages
https://doi.org/10.1155/2018/9461861

Research Article

Discrete Bird Swarm Algorithm Based on Information Entropy Matrix for Traveling Salesman Problem

Min Lin ,[1,2] Yiwen Zhong ,[1,2] Juan Lin,[1] and Xiaoyu Lin[1]

[1]*College of Computer and Information Science, Fujian Agriculture and Forestry University, Fuzhou 350002, China*
[2]*Key Laboratory of Smart Agriculture and Forestry (Fujian Agriculture and Forestry University), Fujian Province University, Fuzhou, Fujian Province 350002, China*

Correspondence should be addressed to Min Lin; linmin@fafu.edu.cn and Yiwen Zhong; yiwzhong@fafu.edu.cn

Received 31 July 2018; Revised 26 September 2018; Accepted 17 October 2018; Published 30 October 2018

Academic Editor: Francesco Riganti-Fulginei

Although bird swarm optimization (BSA) algorithm shows excellent performance in solving continuous optimization problems, it is not an easy task to apply it solving the combination optimization problem such as traveling salesman problem (TSP). Therefore, this paper proposed a novel discrete BSA based on information entropy matrix (DBSA) for TSP. Firstly, in the DBSA algorithm, the information entropy matrix is constructed as a guide for generating new solutions. Each element of the information entropy matrix denotes the information entropy from city i to city j. The higher the information entropy, the larger the probability that a city will be visited. Secondly, each TSP path is represented as an array, and each element of the array represents a city index. Then according to the needs of the minus function proposed in this paper, each TSP path is transformed into a Boolean matrix which represents the relationship of edges. Third, the minus function is designed to evaluate the difference between two Boolean matrices. Based on the minus function and information entropy matrix, the birds' position updating equations are redesigned to update the information entropy matrix without changing the original features. Then three TSP operators are proposed to generate new solutions according to the updated information entropy matrix. Finally, the performance of DBSA algorithm was tested on a large number of benchmark TSP instances. Experimental results show that DBSA algorithm is better or competitively outperforms many state-of-the-art metaheuristic algorithms.

1. Introduction

The traveling salesman problem (TSP) is a classical NP-hard problem, which is easily described but difficult to solve, and it is also a simplified form of multiple complex problems in many fields. The aim of TSP is to find the shortest path that visits each city once and then returns to the starting city. For a symmetric TSP, in the case of n cities, any permutation of n cities yields a possibility, i.e., there are $(n\text{-}1)!/2$ possible paths. The easiest approach to find an optimal path is to evaluate all the possible paths then chooses the shortest one. But the time complexity required for this algorithm is $O(n!)$. It is means that there is no known polynomial time algorithms which can guarantee to find the global optimal solution. Therefore, many studies have attempted to propose various methods for solving TSP problems within an acceptable time and a widely used one is the metaheuristic algorithms. With the powerful performance and the ability to find acceptable solutions within an affordable time, metaheuristic algorithms have been gradually become an alternative to traditional optimization methods over the past decades.

In recent years, many metaheuristic algorithms have been proposed to solve the TSP, such as ant colony algorithm (ACO) [1, 2], artificial bee colony algorithm (ABC) [3], genetic algorithm (GA) [4], particle swarm optimization (PSO) [5], cuckoo search algorithm (CS) [6, 7], bat algorithm (BA) [8, 9], firefly algorithm (FA) [10], invasive weed optimization [11], bacterial evolutionary algorithm [12], dynamic multiscale region search algorithm (DMRSA) [13], a dual local search algorithm [14], immune algorithm [15], simulated annealing algorithm [16], and some hybrid algorithms [17–20].

Bird swarm algorithm (BSA) is a new metaheuristic algorithm recently proposed by Meng et al. [21] for continuous optimization problems. BSA is based on the swarm intelligence extracted from the social behaviors and social interactions in bird swarms. Compared to some of metaheuristic algorithms such as PSO, BSA has the advantages of fast convergence and high convergence precision. Due to its excellent performance, BSA and its variants have been applied in a wide range of application, such as optimization of benchmark functions [22], edge-based target detection for unmanned aerial vehicles using competitive BSA [23], microgrid multiobjective operation optimization [24], edge cloud computing service composition based on modified BSA [25], power flow problems [26], parameter estimation for chaotic systems using improved BSA [27], an improved particle filter based on BSA [28], etc. However, so far there is no solution for solving TSP. Although the basic BSA algorithm is simple and easy to implement, applying BSA algorithm to solve combinatorial optimization problems such as TSP is not a simple task.

In order to extend the basic principle of BSA algorithm to solve TSP without changing the characteristics of original algorithm, this paper presented a novel discrete bird swarm algorithm based on information entropy matrix (DBSA) to solve the TSP problems. The DBSA algorithm first constructs an information entropy matrix I where each element $I_{\mathrm{i,j}}$ represents the information entropy to select city j as next visiting city of city i. Each bird of the DBSA algorithm is responsible for a TSP solution represented by an array which stores the visiting sequence of cities, and according to the needs of minus function the solution is converted into a Boolean matrix B, where each element $b_{\mathrm{i,j}}$ represents whether the corresponding edge is in the solution. In DBSA, the minus function is proposed to evaluate the difference between two Boolean matrices. The calculation results of minus function are substituted into the birds' position update equations to update the information entropy matrix iteratively. Finally, guided by the updated information entropy matrix, birds use three TSP operators to produce new solution. The performance of DBSA algorithm was compared with some of recently published metaheuristic algorithms and some of recently improved classical metaheuristic algorithms on a wide range of benchmark TSP instances.

The remaining sections of this paper are organized as follows: Section 2 provides a short description of the basic BSA algorithm, the goal of TSP and the metaheuristics for the TSP. Section 3 presents our DBSA algorithm. Section 4 compares the performance of DBSA algorithm with some other state-of-the-art algorithms on a large number of TSP instances. Finally, in Section 5 we summarize our study.

2. Related Work

This section introduces the principle of BSA algorithm, the TSP, and metaheuristics algorithm for the TSP. Section 2.1 introduces the basic BSA algorithm. Section 2.2 describes the TSP and its goal. Section 2.3 gives a simple survey of state-of-the-art metaheuristic algorithms for the TSP.

2.1. The Principle of Bird Swarm Algorithm. BSA is a novel metaheuristic algorithm for solving optimization applications. It mimics the birds' foraging behavior, vigilance behavior, and flight behavior to solve the global optimization problems. During the process of foraging, each bird searches food according to individual experience and the population's experience. This behavior can be described mathematically as follows:

$$\begin{aligned} x_{i,j}^{t} = x_{i,j}^{t-1} &+ \left(p_{i,j} - x_{i,j}^{t-1}\right) * C * \operatorname{rand}(0,1) \\ &+ \left(g_{best,j} - x_{i,j}^{t-1}\right) * S * \operatorname{rand}(0,1) \end{aligned} \tag{1}$$

where $x_{\mathrm{i,j}}^{\mathrm{t}}$ denotes the value of the j-th element of the i-th solution at the t-th generation, $\operatorname{rand}(0,1)$ is a uniform distribution function, $p_{\mathrm{i,j}}$ is the best previous position for the j-th element of the i-th bird, and $g_{\mathrm{best,j}}$ denotes the j-th element of global optimal solution. C and S are two positive numbers, which are called cognitive and social accelerated coefficients, respectively.

When keeping vigilance, each bird would try to move towards the center of the swarm and would inevitably compete with others. The vigilance behavior is shown as follows:

$$\begin{aligned} x_{i,j}^{t} = x_{i,j}^{t-1} &+ \left(mean_{j} - x_{i,j}^{t-1}\right) * A1 * \operatorname{rand}(0,1) \\ &+ \left(p_{i,j} - x_{i,j}^{t-1}\right) * A2 * \operatorname{rand}(0,1) \end{aligned} \tag{2}$$

$$A1 = a1 * \exp\left(-\frac{pFit_i}{SumFit + \varepsilon} * N\right) \tag{3}$$

$$A2 = a2 * \exp\left(\left(\frac{pFit_i - pFit_k}{\left|pFit_i - pFit_k\right| + \varepsilon}\right)\frac{pFit_k * N}{SumFit + \varepsilon}\right) \tag{4}$$

where $k\ (k \neq i)$ is a positive integer, which is randomly chosen between 1 and N. *a1* and *a2* are two positive constants in $[0,2]$, $pFit_{\mathrm{i}}$ denotes the i-th bird's best fitness value, and *sumFit* represents the sum of the swarms' best fitness value. ε, which is used to avoid zero-division error, is the smallest constant in the computer. $Mean_{\mathrm{j}}$ denotes the j-th element of the average position of the whole swarm.

Birds would fly to another location from time to time. When flying to another location, birds may often switch between producing and scrounging. The birds with the highest fitness value would be producers, while the ones with the lowest fitness value would be scroungers. Other birds with fitness values between the highest and lowest fitness values would randomly choose to be producer or scrounger. The flight behaviors of the producers and scroungers can be described mathematically as follows, respectively:

$$x_{i,j}^{t} = x_{i,j}^{t-1} * \left(1 + \operatorname{rand}n(0,1)\right) \tag{5}$$

$$x_{i,j}^{t} = x_{i,j}^{t-1} + FL * \left(x_{k,j}^{t-1} - x_{i,j}^{t-1}\right) * \operatorname{rand}(0,1) \tag{6}$$

where rand $n(0,1)$ is a Gaussian distribution with mean 0 and standard deviation 1, $k \in [0,N]$, $k \neq i$. $FL \in (0,2)$ denotes the probability of the scroungers following the producers to search for food. Consider the individual differences, the

```
(1)  Initialize the parameter values of $N$, $M$, $FQ$, $P$;
(2)  Initialize a population of $N$ birds and evaluate the $N$ individuals' fitness value.
(3)  While ($t < M$)
(4)      If ($t$% FQ $\neq 0$)
(5)          For $i = 1$ to $N$
(6)              If rand$(0, 1) < P$
(7)                  Birds forage for food using Eq. (1)
(8)              Else
(9)                  Birds keep vigilance using Eq. (2)
(10)             End if
(11)         End for
(12)     Else
(13)         Divide the swarm into two parts: producers and scroungers.
(14)         For $i = 1$ to $N$
(15)             If ($i$==producer)
(16)                 Birds flight using Eq. (5) //producer
(17)             Else
(18)                 Birds flight using Eq. (6) //scrounger
(19)             End if
(20)         End for
(21)     End if
(22)     Evaluate the fitness value of the new solution;
(23)     If ($f_{\text{new}} < f_{\text{old}}$)
(24)         $f_{\text{old}} = f_{\text{new}}$;
(25)     Update the global optimal solution;
(26)     $t$++
(27) End While
(28) Output the global optimal solution;
```

Algorithm 1: Framework of the BSA.

FL value of each scrounger would randomly select from 0 to 2. The birds switch to flight behavior every *FQ* time steps. Algorithm 1 describes the implementation of BSA. In Algorithm 1, the parameter N denotes the number of population, M denotes the maximum number of iteration, *FQ* represents the frequency of birds' flight behaviors, and P denotes the foraging probability for food.

2.2. Traveling Salesman Problem. TSP is one of the most famous NP-hard combinatorial optimization problems. Given N cities and the coordinates of each city, then TSP is to find a loop that contains the shortest path of all N cities. A valid TSP path can be represented as a cyclic permutation $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$, where π_i denotes the index of the i-th visiting city and π_{i+1} represents the index of the $(i+1)$-th visiting city. The cost of a permutation (tour) is defined as

$$f(\pi) = \sum_{i=1}^{n-1} D(\pi_i, \pi_{i+1}) + D(\pi_n, \pi_1) \tag{7}$$

where $D(\pi_i, \pi_{i+1})$ represents the Euler distance of the two cities. Assuming that the coordinates of the two cities are (x_i, y_i) and (x_{i+1}, y_{i+1}), then the distance calculation is as shown in

$$D = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \tag{8}$$

2.3. Metaheuristic Algorithms for the TSP. In recent years, many metaheuristic algorithms have been proposed for the TSP. Osaba et al. [8] presented an improved discrete bat algorithm which uses hamming distance to measure the distance between bats, and 2-opt and 3-opt operators are adopted to improve solutions. Saji et al. [9] proposed a novel discrete BA (DBA) where two-exchange crossover operator is used to update solutions and 2-opt operator is used to improve solutions. Zhou et al. [11] proposed a discrete invasive weed optimization algorithm (DIWO). DIWO generates a new TSP solution through two local search operators. One is 3-opt operators; another is an improved complete 2-Opt operator. Ouaarab et al. [6] extended and improved CS (IDCS) by reconstructing its population and introducing a new category of cuckoos so that it can solve combinatorial problems as well as continuous problems. In the IDCS algorithm, the 2-opt move method is used for small perturbations, and large perturbations are made by double-bridge move. Zhou et al. [7] proposed a novel discrete CS (DCS) algorithm, which uses learning operator, "A" operator, and 3-opt operator to accelerate the convergence. Saraei et al. [10] proposed a FA which uses greedy swap to extend searching area. Zhong et al. [3] presented a hybrid discrete artificial bee colony algorithm (HABC) with threshold acceptance criterion. Applying a new solution updating equation, HABC learn from other bees and the features of problem synchronously.

Except for above recently published metaheuristic algorithms, many researches have been improving the classical metaheuristic algorithms and applying them to solve TSP. Escario et al. [1] proposed an ant colony extended algorithm (ACE) which includes self-organization property. This self-organization property is based on task division and an emergent task distribution according to the feedback provided by the results of ants' searches. Ismkhan et al. [2] put forward a new ACO algorithm with three effective strategies including pheromone representation with linear space complexity, new next city selection, and pheromone augmented 2-opt local search. Zhang et al. [13] proposed a DMRSA algorithm using vitality selection for TSP. In the DMRSA algorithm, vitality selection (VS) is proposed as a new modification scheme based on delete-oldest selection for TSP. The evaluation criterion of individuals in VS is the individual-made progress in the local successive generations. This is different from the pure fitness criterion. Mahi et al. [19] presented a hybrid method, which used PSO algorithm, ACO algorithm, and 3-Opt heuristic. The PSO algorithm is used for detecting optimum values of parameters which is adopted for city selection operations in the ACO algorithm. The 3-opt operator is used to further improve the best solution produced by ACO. Kóczy et al. [12] presented a discrete bacterial memetic evolutionary algorithm (DBMEA), which is based on the combination of the bacterial evolutionary algorithm and local search techniques. Ouenniche et al. [14] proposed a dual local search framework, that is, a search method that starts with an infeasible solution, explores the dual space, reduces infeasibility iteratively, and lands in the primal space to deliver a feasible solution. Wang [4] improved the GA with two local optimization strategies for TSP. The first local optimization strategy is the four vertices and three lines inequality, which is applied to the local Hamiltonian paths to generate the shorter Hamiltonian circuits (HC). After the HCs are adjusted with the inequality, the second local optimization strategy is executed to reverse the local Hamiltonian paths with more than 2 vertices, which also generates the shorter HCs.

After analyzing the solution updating schemes used in above algorithms for the TSP, we have found it is very important to redesign new position updating equations for the TSP. The equations of original algorithm are suitable to the continuous optimization problems, in order to handle the combinatorial optimization problems, these equations need to be improved or redesigned. In addition, some strategies should be introduced to generate new solution according to the new position updating equations. Guided by these principles, this paper proposes a DBSA algorithm, which not only redesigns the position updating equations for the TSP, but also retains all the characteristics of the original BSA algorithm.

3. Discrete Bird Swarm Algorithm

This section introduces the main ideas of the DBSA algorithms. Section 3.1 explains the concept of information entropy matrix and the construction steps. Section 3.2 describes the representation of solutions. Section 3.3 presents the new position updating equations. Section 3.4 gives the full description of the operators used by birds. Finally, Section 3.5 introduces the implementation steps of DBSA algorithm in detail.

3.1. Information Entropy Matrix. The concept of information entropy was first introduced by Shannon [29]. For the TSP problem, the information entropy of city *i* to city *j* is expressed as follows:

$$I\left(x_{i,j}\right) = \log_2^{1/p_{ij}} \tag{9}$$

The larger value of $I(x_{\mathrm{i,j}})$, the greater possibility of choosing the path of city i to city j. Here p_{ij} represents the probability of city i to city j, and its calculation formula is expressed as follows:

$$p_{ij} = \frac{Dist\,(i,j)}{\sum_{k=1}^{n} Dist\,(i,k)} \quad (i \neq j, i \neq k) \tag{10}$$

where $Dist(i,j)$ represents the distance between city i and city j. Based on the formula for calculating the information entropy, this paper constructs an information entropy matrix to store the information entropy between any two cities. For example, for $n = 4$, the matrix is shown as follows:

$$X_i = \begin{bmatrix} I\left(x_{1,1}\right) & I\left(x_{1,2}\right) & I\left(x_{1,3}\right) & I\left(x_{1,4}\right) \\ I\left(x_{2,1}\right) & I\left(x_{2,2}\right) & I\left(x_{2,3}\right) & I\left(x_{2,4}\right) \\ I\left(x_{3,1}\right) & I\left(x_{3,2}\right) & I\left(x_{3,3}\right) & I\left(x_{3,4}\right) \\ I\left(x_{4,1}\right) & I\left(x_{4,2}\right) & I\left(x_{4,3}\right) & I\left(x_{4,4}\right) \end{bmatrix} \tag{11}$$

3.2. Representation of Solutions. The representation scheme of solutions is simple. Each bird represents a valid TSP path π; each dimension of the bird denotes a city index. For example, for $n = 4$, the bird {3 2 1 4} indicates that the first visiting city is 3, the second visiting city is 2, and so on. In order to incorporate the information entropy matrix, the bird {3 2 1 4} is converted into a Boolean matrix representing the relation of edges. The matrix is shown in (12), where 1 denotes the edge between the two cities which is selected; 0 stands for the edge which is not selected. Figure 1 depicts the conversion steps from the TSP path to the Boolean matrix. For example, the $bird_i$ is {3 2 1 4}; i.e., the TSP path is 3-2-1-4-3, then the first edge from city 3 to city 2 is selected in step 1, the second edge from city 2 to city 1 is selected in step 2, and so on.

$$Edge\left(Bird_i\right) = Edge\left(\{3\ 2\ 1\ 4\}\right) = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{12}$$

3.3. Improved Position Updating Equations. The position updating equations of basic BSA are designed for continuous optimization problems. For combinatorial optimization problems such as TSP, these equations should be redesigned to be consistent with the characteristics of problem at hand as well as retain the good features of original algorithm. Firstly,

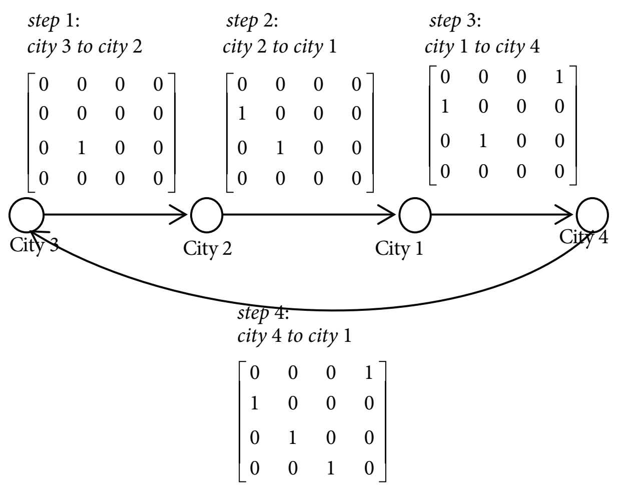


Figure 1: The conversion steps of Boolean matrix.

each TSP solution is converted to an edge matrix according to (12), and then a new minus function is introduced to evaluate the difference between two solutions in the DBSA algorithm. For example, suppose that $bird_i\{1,2,3,4\}$, $p_i\{2,3,1,4\}$, and $g_{best}\{1\ 2\ 4\ 3\}$ represent the i-th bird's solution, the i-th bird's best solution, and the global best solution, respectively. And their corresponding edge matrices are expressed as follows:

$$Edge(Bird_i) = Edge(\{1\ 2\ 3\ 4\}) = \begin{bmatrix} 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1\\ 1&0&0&0 \end{bmatrix}$$

$$Edge(p_i) = Edge(\{2\ 3\ 1\ 4\}) = \begin{bmatrix} 0&0&0&1\\ 0&0&1&0\\ 1&0&0&0\\ 0&1&0&0 \end{bmatrix}$$

$$Edge(g_{best}) = Edge(\{2\ 4\ 3\ 1\}) = \begin{bmatrix} 0&1&0&0\\ 0&0&0&1\\ 1&0&0&0\\ 0&0&1&0 \end{bmatrix} \tag{13}$$

The minus function is shown in (14) and (16), where ϕ denotes null set. In our algorithm, ϕ is set to 0. Equation (15) describes the calculation method for the subtraction of the corresponding elements of the two matrices in the minus function. Let $c_{i,j}$ represent the element of one of matrices in the minus function, and $c'_{i,j}$ denotes the element of another matrix. When $c_{i,j} = 1$, it means that there is an edge from city i to city j. The result of $c_{i,j} - c'_{i,j}$ is shown in (15).

$$\begin{aligned} &Minus(p_i, bird_i) = Minus(Edge(p_i), Edge(bird_i)) \\ &= \begin{bmatrix} 0&0&0&1\\ 0&0&1&0\\ 1&0&0&0\\ 0&1&0&0 \end{bmatrix} - \begin{bmatrix} 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1\\ 1&0&0&0 \end{bmatrix} = \begin{bmatrix} 0&0&0&1\\ 0&0&\phi&0\\ 1&0&0&0\\ 0&1&0&0 \end{bmatrix} \end{aligned} \tag{14}$$

$$c_{i,j} - c'_{i,j} = \begin{cases} 0 & \left(c_{i,j} = 1 \text{ and } c'_{i,j} = 1\right) \text{ or } c_{i,j} = 0 \\ 1 & \text{otherwise} \end{cases} \tag{15}$$

$$\begin{aligned} &Minus(g_{best}, bird_i) \\ &\quad = Minus(Edge(g_{best}), Edge(bird_i)) \\ &= \begin{bmatrix} 0&1&0&0\\ 0&0&0&1\\ 1&0&0&0\\ 0&0&1&0 \end{bmatrix} - \begin{bmatrix} 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1\\ 1&0&0&0 \end{bmatrix} = \begin{bmatrix} 0&\phi&0&0\\ 0&0&0&1\\ 1&0&0&0\\ 0&0&1&0 \end{bmatrix} \end{aligned} \tag{16}$$

Based on (14) and (16), (1) is converted to (17); (2) is converted to (18), where R denotes rand(0, 1). Equations (5) and (6) are converted into (19) and (20), respectively.

$$\begin{aligned} X_i^t &= X_i^{t-1} + Minus(p_i, bird_i) * C * \text{rand}(0,1) + Minus(g_i, bird_i) * S * \text{rand}(0,1) \\ &= \begin{bmatrix} I(x_{1,1}) & I(x_{1,2}) & I(x_{1,3}) & I(x_{1,4}) \\ I(x_{2,1}) & I(x_{2,2}) & I(x_{2,3}) & I(x_{2,4}) \\ I(x_{3,1}) & I(x_{3,2}) & I(x_{3,3}) & I(x_{3,4}) \\ I(x_{4,1}) & I(x_{4,2}) & I(x_{4,3}) & I(x_{4,4}) \end{bmatrix} + \begin{bmatrix} 0&0&0&1\\ 0&0&\phi&0\\ 1&0&0&0\\ 0&1&0&0 \end{bmatrix} * C * \text{rand}(0,1) + \begin{bmatrix} 0&\phi&0&0\\ 0&0&0&1\\ 1&0&0&0\\ 0&0&1&0 \end{bmatrix} * S * \text{rand}(0,1) \\ &= \begin{bmatrix} I(x_{1,1}) & I(x_{1,2}) & I(x_{1,3}) & I(x_{1,4}) + C * R \\ I(x_{2,1}) & I(x_{2,2}) & I(x_{2,3}) & I(x_{2,4}) + S * R \\ I(x_{3,1}) + C * R + S * R & I(x_{3,2}) & I(x_{3,3}) & I(x_{3,4}) \\ I(x_{4,1}) & I(x_{4,2}) + C * R & I(x_{4,3}) + *S * R & I(x_{4,4}) \end{bmatrix} \end{aligned} \tag{17}$$

$$X_i^t = X_i^{t-1} + Minus(Mean_i, bird_i) * A1 * \text{rand}(0,1) + Minus(p_{best}, bird_i) * A2 * \text{rand}(0,1) \tag{18}$$

$$X_i^t = X_i^{t-1} + Edge\left(bird_i\right) * \text{rand}\,(0, 1) \tag{19}$$

$$X_i^t = X_i^{t-1} + FL * Minus\left(bird_k, bird_i\right) * \text{rand}\,(0, 1) \tag{20}$$

3.4. TSP Operators Used by DBSA. After updating the birds' positions, i.e., after obtaining the new information entropy matrix, how to apply the information entropy matrix to generate new TSP path is an essential task to be solved. Therefore, swap, insert, and reverse operators are used to perturb the old TSP path according to the information entropy matrix to generate a new TSP path. Here this section describes the three operators by taking the information entropy matrix of four cities as an example.

$$X_i = \begin{bmatrix} 0 & 5.3 & 0.9 & 4.2 \\ 6.1 & 0 & 3.8 & 1.5 \\ 2.9 & 3.4 & 0 & 0.8 \\ 1.8 & 8.4 & 2.6 & 0 \end{bmatrix} \tag{21}$$

(1) Insert Operator. For $n = 4$, assume that the information entropy matrix is shown in (21), and the i-th bird's solution is {2 3 1 4}. First, randomly select the city 2, i.e., the second row of the matrix. Then randomly select city 1 from the top m cities based on the city's information entropy ranking and insert city 1 behind city 2. The reason why we randomly select a city from the top m cities based on the city's information entropy ranking is that the next visiting city of a city is generally selected from the m cities closer to it. It is represented here as m cities with a large amount of information entropy. Thus, the solution is updated as {2 1 3 4}.

(2) Swap Operator. Similar to the insert operator, for example, the third row of the matrix is randomly selected first and the city 2 in the third row is randomly selected according to its entropy value. Then city 3 is swapped with city 2 to generate a new solution {3 1 2 4}.

(3) Reverse Operator. In the reverse operator, the approach of selecting city is the same as insert and swap operator. The reverse operator refers to the reverse permutation of all cities between two cities. For example, for the $bird_i$\{3 *1* 2 4 *5* 6\}, the result of reverse operator between city 1 and city 5 is {3 *5* 4 2 *1* 6}.

In our DBSA algorithm, the three operators are performed simultaneously, and the optimal operator is selected according to the fitness values. In this way, the optimal solution can be approached more quickly, and the diversity of the solutions is also maintained during the iteration. Although comparing three operators costs extra fitness evaluations, it prevents from falling the local optimum. The detailed implementation process of using the three operators is given in Algorithm 3.

3.5. Implementation of DBSA. Algorithm 2 describes the implementation steps of the DBSA algorithm and Figure 2 draws the flow chart of the DBSA algorithm, where the purpose of perform TSP operators (code (12) and code (22)) in Algorithm 2 is to produce new solution according to the updated information entropy matrix. Algorithm 3 gives the pseudocode of perform TSP operators.

4. Experiments and Discussion

To evaluate the performance of our proposed algorithm, this section compares DBSA algorithm with some of state-of-the-art metaheuristic algorithms on a large number of TSP instances. These TSP instances are selected from the TSPLIB standard library, with a city size ranging from 48 to 33810. Section 4.1 gives the explanation of various parameters in detail and analyzes the algorithm's time complexity. Section 4.2 compares the DBSA algorithm with some of recently published metaheuristic algorithms and Section 4.3 compares it with some of recently improved classic metaheuristic algorithms.

4.1. Parameter Setting and Time Complexity. The DBSA algorithm was implemented with C++ on Visual Studio 2013. The experimental environment was Intel Core 2. 40GHz CPU, 8GB memory, Window 7 OS. Table 1 summarizes the various parameter values used for DBSA. Unless explicitly explained, in all of the following experiments, the maximum iteration number was 2000, the swarm size was 30, and each TSP instance was run 20 times independently.

In addition, in all of the tables below, the column "Best" denotes the best solutions obtained by each algorithm, the column "Worst" represents the worst solutions found, the column "Avg" indicates the average solution length, the column "PA" denotes the percentage error of average solutions, the column "PB" stands for the percentage error of best solutions, and the column "time" shows the average running time in seconds. The "PA" and "PB" values are calculated as follows:

$$PA = \frac{Avg - Opt}{Opt} * 100\% \tag{22}$$

$$PB = \frac{Best - Opt}{Opt} * 100\% \tag{23}$$

The time complexity of the DBSA algorithm is $O(m * n)$, where m represents the population size and n represents the number of iterations. Although the time complexity of DBSA algorithm is similar with other algorithms, the actual search performance is affected by the search strategies of each algorithm, so the actual search performances of each algorithm are quite different. In this section, we compare the performance of DBSA with other algorithms on the basis of similar $m * n$ values and compare the running time of DBSA algorithm with some of algorithms based on similar machine performance.

```
(1) Read the TSP library and constructs the information entropy matrix;
(2) Initialize the values of parameter $N$, $M$, $FQ$, $P$;
(3) Initialize a population of $N$ birds; each bird deals with a TSP path; evaluates the
    length of TSP paths, i.e., the fitness values of birds;
(4) While ($t < M$)
(5)     If ($t$ % FQ $\neq 0$) //foraging or keep vigilance
(6)         For $i = 1$ to $N$
(7)             If rand(0, 1) $< P$
(8)                 Birds forage for food using Eq. (17)
(9)             Else
(10)                Birds keep vigilance using Eq. (18)
(11)            End if
(12)            Perform operator;
(13)        End for
(14)    Else        //flying
(15)        Divide the swarm into two parts: producers and scroungers.
(16)        For $i = 1$ to $N$
(17)            If ($i$==producer)
(18)                Birds flight using Eq. (19)
(19)            Else
(20)                Birds flight using Eq. (20)
(21)            End if
(22)        Perform operator;
(23)        End for
(24)    End if
(25)    $t$++
(26) End While
(27) Output the global optimal solution;
```

Algorithm 2: The framework of DBSA.

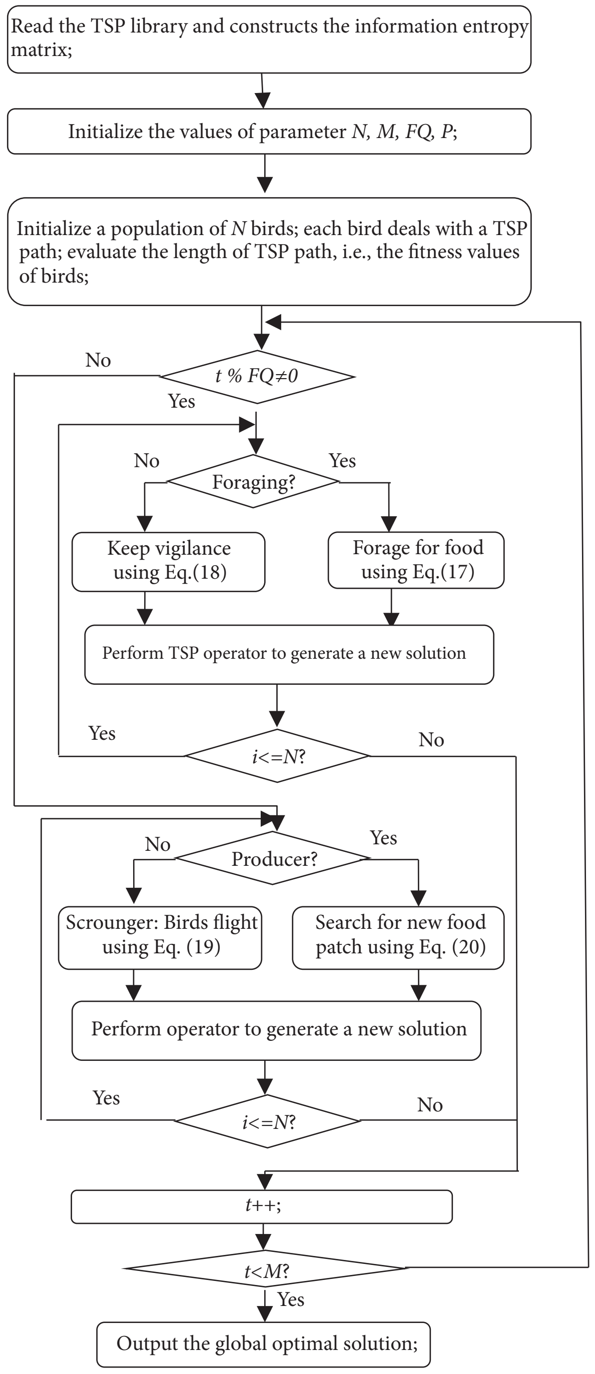
```
(1) Randomly select city $ci$, and find its corresponding row in the information entropy matrix;
(2) Randomly select another city $nci$ from the top $m$ cities based on the values of information entropy;
(3) Perform reverse operator on city $ci$ and city $nci$, and evaluate its fitness value $rvslength$;
(4) Perform swap operator on city $ci$ and city $nci$, and evaluate its fitness value $swplength$;
(5) Perform insert operator on city $ci$ and city $nci$, and evaluate its fitness value $inslength$;
(6) Select the best fitness value as the new solution from the three scheme;
(7) If($Fit_{\text{new}} < Fit_{\text{old}}$)
(8)     $Fit_{\text{new}} = Fit_{\text{old}}$
(9) Update the global best solution;
```

Algorithm 3: Perform TSP operators.

Table 1: Values of various parameters.

Parameter	Value	Parameter meaning
N	30	The number of birds
M	2000	The maximum iteration number
P	[0.8, 1]	The probability of foraging for food
FQ	3	The frequency of birds' flight behaviours

4.2. Compare with Some Recently Published Metaheuristic Algorithms. In order to validate the performance of DBSA algorithm among metaheuristic algorithms, DBSA algorithm is first compared with several newly published metaheuristics, such as DBA [9], IDCS [6], and HABC [3] algorithms. Table 2 gives the comparison results of DBSA with IDCS and DBA algorithms on 41 TSP instances taken from Ouaarab et al. [6] and Saji et al. [9]. The number in the instance title denotes the city size. Among the 41 TSP instances, the minimum size of the city is 51 and the maximum size is 1379. They are all symmetric TSP problems. Each instance was run independently for 20 times. In the IDCS algorithm, the number of cuckoos was 30, the maximum number of iterations was set to 500, and the experiments were conducted on a laptop with Intel Core TM 2 Duo 2.00 GHz CPU and 3 GB of RAM. In the DBA algorithm, the number of bats was 15, the maximum number of iterations was 200, and the experiments were made on a PC with Intel Core 2 Duo 2.1GHZ CPU and 2GB RAM. Among the 41 instances, DBSA algorithm can find the optimal solution on 31 instances, the DBA also found 31 optimal solutions, and IDCS found the



Read the TSP library and constructs the information entropy matrix;
Initialize the values of parameter N, M, FQ, P;
Initialize a population of N birds; each bird deals with a TSP path; evaluate the length of TSP path, i.e., the fitness values of birds;
t % FQ≠0
No
Yes
Foraging?
Keep vigilance using Eq.(18)
Forage for food using Eq.(17)
Perform TSP operator to generate a new solution
i<=N?
Producer?
Scrounger: Birds flight using Eq. (19)
Search for new food patch using Eq. (20)
Perform operator to generate a new solution
t++;
t<M?
Output the global optimal solution;

Figure 2: Implementation step of DBSA algorithm.

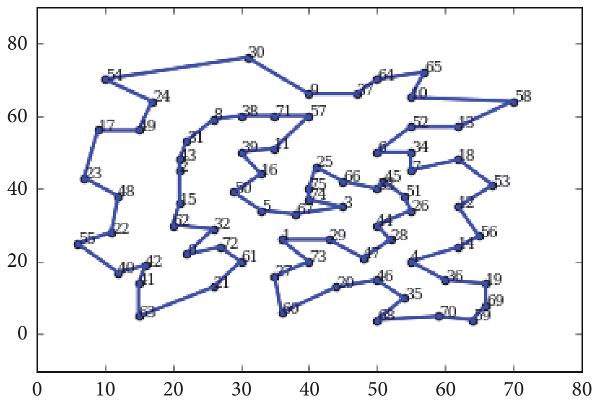


Figure 3: Roadmap for instance eil76 obtained by DBSA.

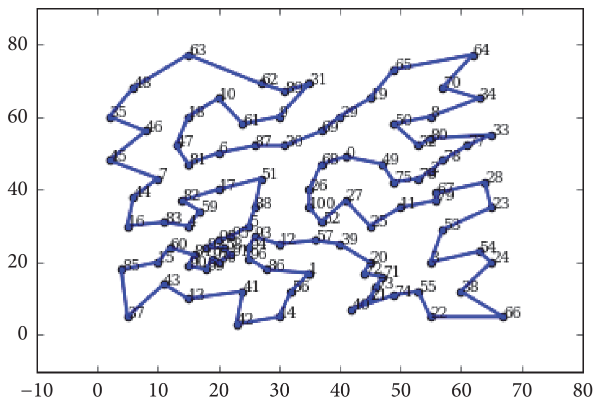


Figure 4: Roadmap for instance eil101 obtained by DBSA.

optimal solution only on 27 instances. The average *PA* values for the DBSA, DBA, and IDCS algorithms are 0.18, 0.45, and 0.60, respectively, which means that the DBSA algorithm is the most stable. When the city size is small, the average solutions found by the three algorithms are similar, but when the city size is greater than 150, the *PA* values, best values, and average values obtained by DBSA algorithm are all superior to IDCS and DBA algorithms.

Table 3 compares the running time of the DBSA algorithm with the IDCS algorithm and the DBA algorithm. The three algorithms run on different machines, but the performances of the machines were similar. Observed from Table 3 that the solving speed of the DBSA algorithm is much faster than the other two algorithms. When the city size reaches 1379, IDCS algorithm takes about one hour, DBA takes about half an hour, and the DBSA algorithm still takes only 15 seconds. The average running time of DBSA, IDCS, and DBA was 1.74, 205.97, and 139.08 second, respectively. Due to the slower solution speed, the city size that IDCS and DBA algorithms can solve is very limited. Moreover, Figures 3, 4, and 5 give the roadmap of instances eil76, eil101, and ch150 obtained by DBSA algorithm, respectively. The points in the roadmaps denote the city index number. These roadmaps further prove the effectiveness of the proposed method.

Table 4 shows the results of the comparison between the DBSA and HABC algorithms. HABC algorithm run on a 2.83 GHz PC with 2GB of RAM, and the performance of the machine is better than DBSA. From the experimental results, DBSA always found 3 optimal solutions, while HABC found 2. The average *PA* values of DBSA and HABC algorithms are 0.64 and 0.65, respectively. The average runtime of DBSA and HABC is 98.20 and 116.17, respectively. Therefore, the

Table 2: Compare DBSA with DBA [9] and IDCS [6] algorithm on 41 TSP instances.

No.	Instance	Opt	DBA				IDCS				DBSA			
			Best	Avg	PB	PA	Best	Avg	PB	PA	Best	Avg	PB	PA
1	eil51	426	426	426	**0.00**	**0.00**	426	426	**0.00**	**0.00**	426	426.2	**0.00**	0.05
2	berlin52	7542	7542	7542	**0.00**	**0.00**	7542	7542	**0.00**	**0.00**	7542	7542	**0.00**	**0.00**
3	st70	675	675	675	**0.00**	**0.00**	675	675	**0.00**	**0.00**	675	675.25	**0.00**	0.04
4	eil76	538	538	538.76	**0.00**	0.14	538	538.03	**0.00**	**0.00**	108159	108293.7	**0.00**	0.12
5	pr76	108,159	108,159	108,159	**0.00**	**0.00**	108159	108159	**0.00**	**0.00**	538	538	**0.00**	**0.00**
6	kroA100	21,282	21,282	21,282	**0.00**	**0.00**	21282	21282	**0.00**	**0.00**	21282	21286.6	**0.00**	0.02
7	kroB100	22,141	22,141	22,141	**0.00**	**0.00**	22141	22141.53	**0.00**	**0.00**	22141	22215.7	**0.00**	0.34
8	kroC100	20,749	20,749	20,753.36	**0.00**	0.02	20749	20749	**0.00**	**0.00**	20749	20749	**0.00**	**0.00**
9	kroD100	21,294	21,294	21,303.50	**0.00**	0.04	21294	21304.33	**0.00**	0.04	21294	21302.75	**0.00**	0.04
10	kroE100	22,068	22,068	22,080.76	**0.00**	0.05	22068	22081.26	**0.00**	0.06	22068	22089.05	**0.00**	0.10
11	eil101	629	629	632.43	**0.00**	0.54	629	630.43	**0.00**	0.22	629	629	**0.00**	**0.00**
12	lin105	14,379	14,379	14,379	**0.00**	**0.00**	14379	14379	**0.00**	**0.00**	14379	14379	**0.00**	**0.00**
13	pr107	44,303	44,303	44,360.8	**0.00**	0.13	44303	44307.06	**0.00**	**0.00**	44303	44368.35	**0.00**	0.15
14	pr124	59,030	59,030	59,037.66	**0.00**	0.012	59030	59030	**0.00**	**0.00**	59030	59035.7	**0.00**	0.01
15	bier127	118,282	118,282	118,385.66	**0.00**	0.08	118282	118359.63	**0.00**	0.06	118282	118302.9	**0.00**	0.02
16	ch130	6110	6110	6124.1	**0.00**	0.23	6110	6135.96	**0.00**	0.42	6110	6123	**0.00**	0.21
17	pr136	96,772	96,772	96,995	**0.00**	0.23	96790	97009.26	0.01	0.24	96772	96817.7	**0.00**	0.05
18	pr144	58,537	58,537	58,537	**0.00**	**0.00**	58537	58537	**0.00**	0.00	58537	58552.9	**0.00**	0.03
19	ch150	6528	6528	6550.3	**0.00**	0.34	6528	6549.9	**0.00**	0.33	6528	6538.6	**0.00**	0.16
20	kroA150	26,524	26,524	26,560.2	**0.00**	0.13	26524	26569.26	**0.00**	0.17	26524	26534.1	**0.00**	0.04
21	kroB150	26,130	26,130	26,146.63	**0.00**	0.06	26130	26159.3	**0.00**	0.11	26130	26137.25	**0.00**	0.03
22	pr152	73,682	73,682	73,759.06	**0.00**	0.10	73682	73682	**0.00**	0.00	73682	73716	**0.00**	0.05
23	rat195	2323	2324	2340.7	**0.04**	0.76	2324	2341.86	0.04	0.81	2328	2330.05	0.22	0.30
24	d198	15,780	15,780	15,802.83	**0.00**	0.14	15781	15807.66	**0.00**	0.17	15780	15788.05	**0.00**	0.05
25	kroA200	29,368	29,368	29,449.23	**0.00**	0.27	29382	29446.66	0.04	0.26	29368	29392.05	**0.00**	0.08
26	kroB200	29,437	29,439	29,527.4	**0.00**	0.30	29448	29542.49	0.03	0.29	29437	29451.15	**0.00**	0.05
27	ts225	126,643	126,643	126,643	**0.00**	**0.00**	126643	126659.23	**0.00**	0.01	126643	126643	**0.00**	**0.00**
28	tsp225	3916	3916	3944.8	**0.00**	0.73	3916	3958.76	**0.00**	1.09	3919	3932.9	0.08	0.43
29	pr226	80,369	80,369	80,409.1	**0.00**	0.04	80369	80386.66	**0.00**	0.02	80369	80404.4	**0.00**	0.04
30	gil262	2378	2380	2390.7	0.08	0.53	2382	2394.5	0.16	0.68	2378	2379.6	**0.00**	0.07
31	pr264	49,135	49,135	49,167.9	**0.00**	0.06	49135	49257.5	**0.00**	0.24	49135	49135	**0.00**	**0.00**
32	a280	2579	2579	2611,	**0.00**	0.30	2579	2592.33	**0.00**	0.51	2579	2579	**0.00**	**0.00**
33	pr299	48,191	48,191	48,311.7	**0.00**	0.25	48207	48470.53	0.03	0.58	48191	48241.55	**0.00**	0.10
34	lin318	42,029	42,154	42,462.16	0.29	1.03	42125	42434.73	0.22	0.96	42061	42209.7	0.08	0.43
35	rd400	15,281	15,336	15,465.3	0.35	1.20	15447	15533.73	1.08	1.65	15301	15329.75	0.13	0.32
36	fl417	11,861	11,865	11,884.1	0.03	0.19	11873	11910.53	0.1	0.41	11878	11929.45	0.14	0.58
37	pr439	107,217	107,291	107,683.33	0.06	0.43	107447	107960.5	0.21	0.69	107285	107369.6	0.06	0.14
38	rat575	6773	6862	6903.83	1.31	1.93	6896	6956.73	1.81	2.71	6810	6828.1	0.55	0.81
39	rat783	8806	8948	9010.4	1.61	2.32	9043	9109.26	2.69	3.44	8836	8873.9	0.34	0.77
40	pr1002	259,045	266,146	266,412.8	2.74	2.84	266508	268630.03	2.88	3.70	260687	261339.9	0.63	0.89
41	nrw1379	56,638	58,188	58,299	2.73	2.93	58951	59349.53	4.08	4.78	57024	57126.2	0.68	0.86

performance of DBSA algorithm is slightly better than HABC algorithm.

4.3. Compare with Some Recently Improved Classical Metaheuristics. In order to further observe the performance of DBSA algorithm and give more credibility of our improvement, DBSA algorithm was compared with several recently improved classical metaheuristic algorithms, such as DBMEA [12], DMRSA [13], ACE [1], and HGA [4] algorithms. Table 5 gives the results of comparison between DBSA and ACE algorithm on 22 TSP instances, where Ry48, Ftv70, Ftv170, and Kro124p are asymmetric TSP problems. ACE is an extended ant colony algorithm. The maximum number of iterations of the ACE algorithm is $400 * k$,

Table 3: The running time of DBA [9], IDCS [6], and DBSA algorithm.

No	Instance	DBA	IDCS	DBSA
1	Eil51	0.20	1.16	0.16
2	Berlin52	0.03	0.09	0.02
3	St70	0.43	1.56	0.21
4	Pr76	0.57	4.73	0.11
5	Eil76	1.54	6.54	0.10
6	KroA100	1.36	2.70	0.21
7	KroB100	3.35	8.74	0.49
8	KroC100	2.51	3.36	0.21
9	KroD100	7.55	8.35	0.38
10	KroE100	11.12	14.18	0.53
11	Eil101	17.09	18.74	0.31
12	Lin105	2.27	5.01	0.25
13	Pr107	18.01	12.89	0.43
14	Pr124	2.57	3.36	0.17
15	Bier127	19.14	25.50	0.50
16	Ch130	13.68	23.12	0.51
17	Pr136	22.10	35.82	0.69
18	Pr144	2.12	2.96	0.58
19	Ch150	25.70	27.74	0.79
20	KroA150	21.75	31.23	0.78
21	KroB150	22.17	33.01	0.81
22	Pr152	15.24	14.86	0.64
23	Rat195	42.30	57.25	1.06
24	D198	38.75	59.95	1.16
25	KroA200	46.97	62.08	1.12
26	KroB200	53.10	64.06	1.13
27	Ts225	18.24	47.51	0.55
28	Tsp225	80.61	76.16	1.30
29	Pr226	44.89	50.00	1.04
30	Gil262	81.25	102.39	1.58
31	Pr264	64.51	82.93	1.37
32	A280	28.6	115.57	1.37
33	Pr299	102.64	138.20	1.84
34	Lin318	120.14	156.17	2.04
35	Rd400	194.11	264.94	2.81
36	Fl417	112.36	274.59	3.75
37	Pr439	223.09	308.75	3.22
38	Rat575	423.56	506.67	4.07
39	Rat783	758.49	968.66	6.52
40	Rr1002	1195.20	1662.61	9.94
41	Nrw1379	1863.12	3160.47	16.52

where k denotes the city size, while the maximum number of iterations of the DBSA is 2000, $2000 < 400 * k$. Among the 22 TSP instances, DBSA and ACE both found 19 optimal solutions. And DBSA always found 2 optimal solutions, while ACE always found 0. The average *PA* values found by DBSA and ACE are 0.29 and 0.59, respectively. The average *PB* values of DBSA and ACE are 0.06 and 0.09, respectively. The average solutions obtained by DBSA are always better than ACE, and the worst solutions found by DBSA are also better than ACE. Therefore, the performance of DBSA is significantly better than ACE. Table 6 compares DBSA algorithm with DBMEA algorithm on 14 symmetric VLSI TSP benchmark problems which are taken from [12]. The number of bacteria was 100 in the DBMEA algorithm. From the experimental results, the average *PB* values of DBSA and DBMEA are 1.43 and 0.36, respectively. The average *PA* values of DBSA and DBMEA are 1.48 and 0.62, respectively. Therefore, regardless of the optimal solutions or the average

Table 4: Comparison between DBSA algorithm and HABC [3] algorithm on 39 TSP instances.

No	Instance	Opt	HABC		DBSA	
			PA	Time	PA	Time
1	Ch150	6528	0.31	1.86	0.20	1.19
2	Kroa150	26,524	0.05	1.88	0.02	1.21
3	Krob150	26,130	-0.01	1.82	0.06	1.21
4	Pr152	73,682	**0.00**	1.89	0.10	0.87
5	U159	42,080	-0.01	2.01	**0.00**	0.44
6	Rat195	2323	0.61	2.18	0.32	1.88
7	D198	15,780	0.27	2.28	0.05	1.80
8	Kroa200	29,368	0.05	2.4	0.06	1.49
9	Krob200	29,437	0.02	2.31	0.02	1.41
10	Ts225	126,643	0.00	2.78	**0.00**	1.11
11	Pr226	80,369	0.00	2.84	0.03	1.72
12	Gil262	2378	0.38	3.42	0.07	2.33
13	Pr264	49,135	**0.00**	2.85	**0.00**	1.29
14	Pr299	48,191	0.11	3.7	0.07	2.67
15	Lin318	42,029	0.26	3.52	0.40	3.10
16	Rd400	15281	0.26	4.98	0.28	4.23
17	Fl417	11,861	1.01	5.61	0.61	4.75
18	Pr439	107,217	0.22	5.68	0.14	4.53
19	Pcb442	50,778	0.15	5.93	0.33	4.56
20	U574	36,905	0.37	8.85	0.83	6.05
21	Rat575	6773	0.75	8.83	0.77	6.08
22	U724	41,910	0.33	13.43	0.62	8.82
23	Rat783	8806	0.91	15.29	0.76	8.80
24	Pr1002	259,045	0.71	11.19	0.83	12.54
25	Pcb1173	56,892	0.77	14.67	0.88	14.66
26	D1291	50,801	1.64	14.32	0.79	16.01
27	Rl1323	270,199	0.5	15.29	0.69	17.23
28	Fl1400	20,127	1.29	18.16	1.76	32.46
29	D1655	62,128	1.28	21.28	1.01	22.31
30	Vm1748	336,556	0.72	25.21	0.67	26.19
31	U2319	234,256	0.26	25.54	0.38	33.22
32	Pcb3038	137,694	1.03	40.42	0.84	45.06
33	Fnl4461	182,566	1.30	44.21	1.14	89.86
34	Rl5934	556,045	1.79	63.74	1.09	135.28
35	Pla7397	23,260,728	1.47	108.34	2.38	300.23
36	Usa13509	199,82,859	1.57	434.04	1.84	563.02
37	Brd14051	468,385	1.67	452.87	1.54	343.29
38	D18512	645,238	1.51	816.38	1.22	446.58
39	Pla33810	66,048,945	1.81	2318.8	2.26	1660.20

solutions, the DBSA algorithm is superior to the DBMEA algorithm.

Table 7 shows the comparison results of the DBSA algorithm and the DMRSA algorithm on 24 TSP instances [13]. The maximum number of iterations of the DBSA algorithm is 2000, but DMRSA uses a local search strategy, which requires more internal iterations. The numbers of optimal solutions found by DMRSA and DBSA algorithms are both 17. The average *PB* values of DMRSA and DBSA algorithms are 0.37 and 0.06, respectively. The average *PA* values of DMRSA and DBSA algorithms are 0.65 and 0.22, respectively. When the city size is less than 200, the performance of DBSA and DMRSA is relatively close. When the city size is greater than 200, the performance of DBSA is significantly better than DMRSA. Table 8 compares the running time of DMRSA and DBSA on 10 TSP instances. As can be seen from the table, the average running time of the DMRSA algorithm is 50 times that of the DBSA algorithm. Therefore, the performance of DBSA algorithm is superior to DMRSA algorithm.

Table 5: Comparison between DBSA algorithm and ACE [1] algorithm.

No.	Instance	Opt	ACE					DBSA				
			Best	Worst	Avg	PB	PA	Best	Worst	Avg	PB	PA
1	Ry48	14422	**14422**	14883	14495.8	**0.00**	0.51	**14422**	14673	14521.65	**0.00**	0.69
2	Eil51	426	**426**	432	426.818	**0.00**	0.19	**426**	427	426.4	**0.00**	0.09
3	Berlin52	7542	**7542**	7715	7543.04	**0.00**	0.01	**7542**	7542	7542	**0.00**	**0.00**
4	Ftv70	1950	**1950**	2060	1968.16	**0.00**	0.93	**1950**	2013	1957.45	**0.00**	0.38
5	St70	675	**675**	691	676.418	**0.00**	0.21	**675**	684	678.9	**0.00**	0.58
6	Eil76	538	**538**	546	538.311	**0.00**	0.06	**538**	538	538	**0.00**	0.00
7	Pr76	108159	**108159**	109085	108251	**0.00**	0.09	**108159**	109186	108210.4	**0.00**	0.05
8	Rat99	1211	**1211**	1246	1213.29	**0.00**	0.19	**1211**	1212	1211.7	**0.00**	0.06
9	Kroa100	21282	**21282**	21600	21298.6	**0.00**	0.08	**21282**	21305	21285.45	**0.00**	0.02
10	Eil101	629	**629**	644	633.619	**0.00**	0.73	**629**	631	629.35	**0.00**	0.06
11	Lin105	14379	**14379**	14514	14385.5	**0.00**	0.05	**14379**	14401	14384.5	**0.00**	0.04
12	Kro124p	36230	**36230**	37777	36460.8	**0.00**	0.64	**36230**	36459	36256.9	**0.00**	0.07
13	Ch130	6110	**6110**	6259	6153.96	**0.00**	0.72	**6110**	6160	6127.9	**0.00**	0.29
14	Ch150	6528	**6528**	6670	6550	**0.00**	0.34	6549	6564	6555.7	0.32	0.42
15	Pr152	73682	**73682**	74802	73766.8	**0.00**	0.12	**73682**	74287	73885.2	**0.00**	0.28
16	U159	42080	**42080**	43816	42199.8	**0.00**	0.28	**42080**	42704	42377.45	**0.00**	0.71
17	Ftv170	2755	**2755**	3007	2824.08	**0.00**	2.51	**2755**	2787	2768.75	**0.00**	0.50
18	D198	15780	**15780**	15899	15813.3	**0.00**	0.21	**15780**	15818	15798.8	**0.00**	0.12
19	Gil262	2378	**2378**	2413	2390.06	**0.00**	0.51	**2378**	2390	2382.6	**0.00**	0.19
20	D493	35002	35123	35770	35449.3	0.35	1.28	35106	35332	35200.45	0.3	0.57
21	Rat783	8806	8868	9010	8936.7	0.70	1.48	8836	8882	8855.2	0.34	0.56
22	Fl1400	20127	20328	21067	20491.7	1.00	1.81	20205	20379	20273.65	0.39	0.73

Table 6: Compare DBSA with DBMEA [12] algorithm.

No	Instances	Opt	DBMEA		DBSA	
			PB	PA	PB	PA
1	xql662	2513	1.51	1.51	0.08	0.35
2	dkg813	3199	1.39	1.39	0.13	0.42
3	dka1376	4666	1.56	1.65	0.32	0.65
4	dca1389	5085	1.40	1.44	0.35	0.69
5	dja1436	5257	1.43	1.50	0.17	0.46
6	icw1483	4416	1.16	1.27	0.18	0.38
7	rbv1583	5387	1.10	1.17	0.45	0.62
8	rby1599	5533	1.01	1.07	0.31	0.63
9	dea2382	8017	1.71	1.80	0.21	0.41
10	pds2566	7643	1.73	1.81	0.31	0.65
11	bch2762	8234	1.73	1.78	0.52	0.80
12	fdp3256	10008	0.96	1.01	0.59	0.81
13	dkc3938	12503	1.70	1.76	0.72	0.89
14	xqd4966	15316	1.56	1.60	0.71	0.88

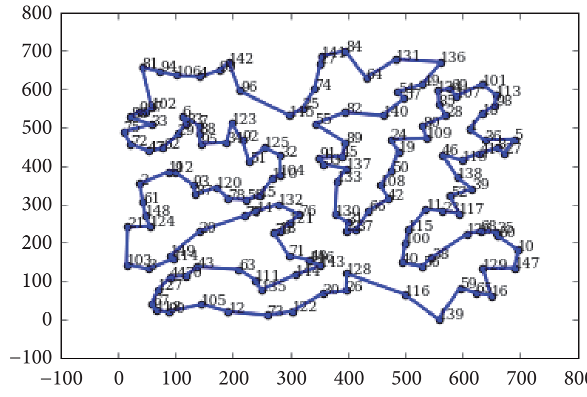


Figure 5: Roadmap for instance ch150 obtained by DBSA.

Table 9 gives the comparison results between DBSA and HGA algorithms. The HGA algorithm uses two local optimization strategies, so that more iterative times are actually needed. The numbers of optimal solutions found by HGA and DBSA algorithms are 9 and 30, respectively. And DBSA always found 9 optimal solutions, while HGA found 0. The average *PB* values of DBSA and HGA are 0.01 and 0.74, respectively. The average *PA* values of DBSA and HGA are 0.10 and 1.05, respectively. So the overall performance of DBSA algorithm outperforms the HGA algorithm.

5. Conclusion and Future Work

BSA algorithm is a novel metaheuristic algorithm inspired from the bird swarms and was first proposed for continuous optimization problems. In order to apply it to solve the combinatorial optimization problem such as TSP, it is necessary to use appropriate strategies to ensure the characteristics of the original algorithm and to design suitable

Table 7: Compare DBSA with DMRSA [13] algorithm.

No	Instance	Opt	DMRSA				DBSA			
			Best	Avg	PB	PA	Best	Avg	PB	PA
1	eil51	426	**426**	426.48	**0.00**	0.11	**426**	426.55	**0.00**	0.13
2	berlin52	7542	**7542**	7542	**0.00**	**0.00**	**7542**	7542	**0.00**	**0.00**
3	eil76	538	**538**	540.36	**0.00**	0.44	**538**	538.15	**0.00**	0.03
4	rd100	7910	**7910**	7912.53	**0.00**	0.03	**7910**	7922	**0.00**	0.15
5	kroA100	21282	**21282**	21282	**0.00**	**0.00**	**21282**	21286.6	**0.00**	0.02
6	kroB100	22141	**22141**	22184.61	**0.00**	0.20	**22141**	22190.2	**0.00**	0.22
7	kroC100	20749	**20749**	20751.14	**0.00**	0.01	**20749**	20759.3	**0.00**	0.05
8	kroD100	21294	**21294**	21296.70	**0.00**	0.01	**21294**	21309	**0.00**	0.07
9	kroE100	22068	**22068**	22114.96	**0.00**	0.21	**22068**	22140.55	**0.00**	0.33
10	eil101	629	**629**	630.98	**0.00**	0.31	**629**	629.3	**0.00**	0.05
11	lin105	14379	**14379**	14379	**0.00**	**0.00**	**14379**	14382.3	**0.00**	0.02
12	bier127	118282	**118282**	118435.63	**0.00**	0.13	**118282**	118353.4	**0.00**	0.06
13	ch130	6110	**6110**	6120.36	**0.00**	0.17	**6110**	6125.15	**0.00**	0.25
14	ch150	6528	**6528**	6547.28	**0.00**	0.30	**6528**	6552.8	**0.00**	0.38
15	kroA150	26524	**26524**	26568.67	**0.00**	0.17	**26524**	26564.6	**0.00**	0.15
16	kroB150	26130	**26130**	26144.85	**0.00**	0.06	**26132**	26148.9	**0.01**	0.07
17	kroA200	29368	**29368**	29439.08	**0.00**	0.24	**29368**	29434.8	**0.00**	0.23
18	kroB200	29437	29438	29578.44	**0.00**	0.48	**29437**	29468.8	**0.00**	0.11
19	lin318	42029	42040	42232.04	0.03	0.48	42119	42259.7	0.21	0.55
20	rat575	6773	6909	6953.225	2.01	2.66	6795	6804	0.32	0.46
21	rat783	8806	9031	9090.37	2.56	3.23	8824	8839.4	0.20	0.38
22	rl1323	270199	271640	274083.42	0.53	1.44	270356	270807.1	0.06	0.23
23	fl1400	20127	20348	20464.31	1.10	1.68	20194	20255.3	0.33	0.64
24	d1655	62128	63847	64202.7	2.77	3.34	62326	62507.65	0.32	0.61

Table 8: The running time of DMRSA [13] and DBSA (in seconds).

instance	DMRSA	DBSA
att48	10.74	0.36
eil51	10.48	0.47
berlin52	11.06	0.03
st70	17.12	0.13
eil76	20.18	0.25
pr76	19.41	0.15
kroA100	29.66	0.64
rd100	32.17	0.69
eil101	29.53	0.67
kroA200	154.57	3.21

schemes according to different combinatorial optimization problems. Based on these principles, this paper presents a novel discrete BSA with information entropy matrix. Guided by the information entropy matrix, a minus function is introduced to evaluate the difference between two solutions, and the position updating equations of birds are redesigned to update the information entropy matrix. Meanwhile, three TSP operators are introduced to produce new solutions according to the information entropy matrix. Experiment results show that these strategies are very efficient for TSP. The performance of DBSA outperforms significantly many metaheuristic algorithms in most of the cases.

In our future research, we will apply the designing principles and the analysis procedure of proposed DBSA algorithm to guide the design and implementation of other metaheuristic algorithms for other discrete optimization problems.

Data Availability

The TSP data used to support the findings of this study have been deposited in the TSPLIB repository (https://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the project of Natural Science Foundation of Fujian Province (no. 2015J01233), the project of Fujian Provincial Department of Education (no.

Table 9: Compare DBSA with HGA [4] algorithm.

No.	Instance	Opt	HGA		DBSA	
			PB	PA	PB	PA
1	Eil51	426	0.67	0.75	**0.00**	0.09
2	Berlin52	7542	0.03	0.03	**0.00**	**0.00**
3	St70	675	0.31	0.35	**0.00**	**0.00**
4	Eil76	538	1.18	1.50	**0.00**	**0.00**
5	Pr76	108159	**0.00**	0.09	**0.00**	0.08
6	Rat99	1211	0.68	0.90	**0.00**	0.01
7	KroA100	21282	0.02	0.14	**0.00**	0.01
8	KorC100	20749	0.01	0.30	**0.00**	**0.00**
9	kroD100	21294	**0.00**	0.24	**0.00**	0.08
10	Rd100	7910	**0.00**	0.04	**0.00**	**0.00**
11	Eil101	629	1.78	2.52	**0.00**	**0.00**
12	Lin105	14379	0.03	0.31	**0.00**	0.02
13	Pr107	44,303	**0.00**	0.09	**0.00**	0.15
14	Pr124	59,030	**0.00**	0.11	**0.00**	**0.00**
15	Ch130	6110	0.01	0.33	**0.00**	0.21
16	Pr136	96772	0.01	0.26	**0.00**	0.12
17	Pr144	58537	**0.00**	0.00	**0.00**	0.03
18	kroA150	26524	**0.00**	0.28	**0.00**	0.02
19	kroB150	26130	-0.01	0.79	**0.00**	0.05
20	Ch150	6528	0.04	0.45	**0.00**	0.15
21	Pr152	73682	**0.00**	0.11	**0.00**	0.10
22	Rat195	2323	1.04	1.42	0.22	0.36
23	D198	15,780	0.74	1.16	**0.00**	0.06
24	kroA200	29,368	**0.00**	0.31	**0.00**	0.05
25	kroB200	29,437	0.05	0.50	**0.00**	0.04
26	Tsp225	3916	-0.95	-0.59	**0.00**	0.39
27	Ts225	126643	1.18	1.30	**0.00**	0.00
28	Pr226	80369	0.08	0.21	**0.00**	0.05
29	Pr264	49135	0.03	0.06	**0.00**	**0.00**
30	A280	2579	3.13	3.77	**0.00**	**0.00**
31	Pr299	48191	2.64	3.25	**0.00**	0.10
32	Lin318	42029	1.42	2.02	0.04	0.45
33	Rd400	15281	5.03	5.65	0.11	0.35
34	Pr439	107217	2.76	3.72	0.05	0.14
35	Pcb442	50778	4.13	4.41	0.07	0.30

JAT160143; no. JAT170181), the special Fund for Scientific and Technological Innovation of Fujian Agriculture and Forestry University (no. CXZX2016026; no. CXZX2016031).

References

[1] J. B. Escario, J. F. Jimenez, and J. M. Giron-Sierra, "Ant colony extended: Experiments on the travelling salesman problem," *Expert Systems with Applications*, vol. 42, no. 1, pp. 390–410, 2015.

[2] H. Ismkhan, "Effective heuristics for ant colony optimization to handle large-scale problems," *Swarm and Evolutionary Computation*, vol. 32, pp. 140–149, 2017.

[3] Y. Zhong, J. Lin, L. Wang, and H. Zhang, "Hybrid discrete artificial bee colony algorithm with threshold acceptance criterion for traveling salesman problem," *Information Sciences*, vol. 421, pp. 70–84, 2017.

[4] Y. Wang, "The hybrid genetic algorithm with two local optimization strategies for traveling salesman problem," *Computers & Industrial Engineering*, vol. 70, no. 1, pp. 124–133, 2014.

[5] M. A. H. Akhand, S. Akter, M. A. Rashid, and S. B. Yaakob, "Velocity tentative PSO: An optimal velocity implementation based particle swarm optimization to solve traveling salesman problem," *IAENG International Journal of Computer Science (IJCS)*, vol. 42, no. 3, pp. 1–12, 2015.

[6] A. Ouaarab, B. Ahiod, and X.-S. Yang, "Discrete cuckoo search algorithm for the travelling salesman problem," *Neural Computing and Applications*, vol. 24, no. 7-8, pp. 1659–1669, 2014.

[7] Y. Zhou, X. Ouyang, and J. Xie, "A discrete cuckoo search algorithm for travelling salesman problem," *International Journal of Collaborative Intelligence*, vol. 1, no. 1, p. 68, 2014.

[8] E. Osaba, X.-S. Yang, F. Diaz, P. Lopez-Garcia, and R. Carballedo, "An improved discrete bat algorithm for symmetric and asymmetric Traveling Salesman Problems," *Engineering Applications of Artificial Intelligence*, vol. 48, pp. 59–71, 2016.

[9] Y. Saji and M. E. Riffi, "A novel discrete bat algorithm for solving the travelling salesman problem," *Neural Computing and Applications*, vol. 27, no. 7, pp. 1853–1866, 2016.

[10] M. Saraei, R. Analouei, and P. Mansouri, "Solving of travelling salesman problem using firefly algorithm with greedy approach," *Cumhuriyet Science Journal*, vol. 36, no. 6, pp. 267–273, 2015.

[11] Y. Zhou, Q. Luo, H. Chen, A. He, and J. Wu, "A discrete invasive weed optimization algorithm for solving traveling salesman problem," *Neurocomputing*, vol. 151, no. 3, pp. 1227–1236, 2015.

[12] L. T. Kóczy, P. Földesi, B. Tüű-Szabó et al., "Enhanced discrete bacterial memetic evolutionary algorithm—an efficacious metaheuristic for the traveling salesman optimization," *Information Sciences*, vol. 460/461, pp. 389–400, 2018.

[13] H. Zhang and J. Zhou, "Dynamic multiscale region search algorithm using vitality selection for traveling salesman problem," *Expert Systems with Applications*, vol. 60, pp. 81–95, 2016.

[14] J. Ouenniche, P. K. Ramaswamy, and M. Gendreau, "A dual local search framework for combinatorial optimization problems with TSP application," *Journal of the Operational Research Society*, vol. 68, no. 11, pp. 1377–1398, 2017.

[15] Z. Xu, Y. Wang, S. Li, Y. Liu, Y. Todo, and S. Gao, "Immune algorithm combined with estimation of distribution for traveling salesman problem," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 11, pp. S142–S154, 2016.

[16] X. Geng, Z. Chen, W. Yang, D. Shi, and K. Zhao, "Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search," *Applied Soft Computing*, vol. 11, no. 4, pp. 3680–3689, 2011.

[17] S. Chen and C. Chien, "Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques," *Expert Systems with Applications*, vol. 38, no. 12, pp. 14439–14450, 2011.

[18] W. Deng, R. Chen, B. He, Y. Liu, L. Yin, and J. Guo, "A novel two-stage hybrid swarm intelligence optimization algorithm and application," *Soft Computing*, vol. 16, no. 10, pp. 1707–1722, 2012.

[19] M. Mahi, Ö. K. Baykan, and H. Kodaz, "A new hybrid method based on particle swarm optimization, ant colony optimization and 3-Opt algorithms for traveling salesman problem," *Applied Soft Computing*, vol. 30, pp. 484–490, 2015.

[20] Y. Zhong, J. Lin, L. Wang, and H. Zhang, "Discrete comprehensive learning particle swarm optimization algorithm with Metropolis acceptance criterion for traveling salesman problem," *Swarm and Evolutionary Computation*, vol. 42, pp. 77–88, 2018.

[21] X. B. Meng, X. Z. Gao, L. Lu, Y. Liu, and H. Zhang, "A new bio-inspired optimisation algorithm: Bird Swarm Algorithm," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 28, no. 4, pp. 673–687, 2016.

[22] M. Parashar, S. Rajput, H. M. Dubey, and M. Pandit, "Optimization of benchmark functions using a nature inspired bird swarm algorithm," in *Proceedings of the 2017 3rd International Conference on Computational Intelligence & Communication Technology (CICT)*, pp. 1–7, Ghaziabad, India, Feburary 2017.

[23] X. Wang, Y. Deng, and H. Duan, "Edge-based target detection for unmanned aerial vehicles using competitive Bird Swarm Algorithm," *Aerospace Science and Technology*, vol. 78, pp. 708–720, 2018.

[24] C. Zeng, C. Peng, K. Wang, Y. Zhang, and M. Zhang, "Multi-objective operation optimization of micro grid based on bird swarm algorithm," *Power System Protection Control*, vol. 44, no. 13, pp. 117–122, 2016.

[25] C. Jian, M. Li, and X. Kuang, "Edge cloud computing service composition based on modified bird swarm optimization in the internet of things," *Cluster Computing*, vol. 12, pp. 1–9, 2018.

[26] M. Ahmad, N. Javaid, I. A. Niaz, S. Shafiq, O. U. Rehman, and H. M. Hussain, "Application of Bird Swarm Algorithm for Solution of Optimal Power Flow Problems," in *Complex, Intelligent, and Software Intensive Systems*, vol. 772 of *Advances in Intelligent Systems and Computing*, pp. 280–291, Springer International Publishing, Cham, 2019.

[27] C. Xu and R. Yang, "Parameter estimation for chaotic systems using improved bird swarm algorithm," *Modern Physics Letters B. Condensed Matter Physics, Statistical Physics, Applied Physics*, vol. 31, no. 36, 15 pages, 2017.

[28] L. Zhang, Q. Bao, W. Fan, K. Cui, H. Xu, and Y. Du, "An Improved Particle Filter Based on Bird Swarm Algorithm," in *Proceedings of the 2017 10th International Symposium on Computational Intelligence and Design (ISCID)*, pp. 198–203, Hangzhou, December 2017.

[29] C. E. Shannon, "A mathematical theory of communication," *Bell Labs Technical Journal*, vol. 27, pp. 379–423, 1948.