A novel genetic algorithm for large scale colored balanced traveling salesman problem

Xueshi Dong, Yongle Cai

PII: S0167-739X(18)31728-X
DOI: https://doi.org/10.1016/j.future.2018.12.065
Reference: FUTURE 4686

To appear in: *Future Generation Computer Systems*

Received date : 22 July 2018
Revised date : 20 October 2018
Accepted date : 29 December 2018

A Novel Genetic Algorithm for Large Scale Colored Balanced Traveling Salesman Problem

Xueshi Dong[a, b], Yongle Cai[b]

[a] College of Computer Science and Technology, Qingdao University, Qingdao 266071, China

[b] Computer School, Wuhan University, Wuhan 430072, China

Abstract—The paper gives an applicable model called colored balanced traveling salesman problem (CBTSP), it is utilized to model optimization problems with partially overlapped workspace such as the scheduling and deploying of the resources and goods. CBTSP is NP-hard problem, the traditional nature-inspired algorithms, such as genetic algorithm (GA), hill-climbing GA and simulated annealing GA, are easy to fall into local optimum. In order to improve it, the paper proposes a novel genetic algorithm (NGA) based on ITÖ process to solve CBTSP. First of all, NGA utilizes the dual-chromosome coding to represent solution of this problem, and then updates the solution by the crossover and mutation operator. During the process of crossover operator, the length of crossover can be affected by activity intensity, which is directly proportional to environmental temperature and inversely proportional to particle radius. The experiments verify that NGA can demonstrate better solution quality than the compared algorithms for large scale CBTSP.

Key words—Novel genetic algorithm, Large scale optimization, Colored balanced traveling salesman problem, Colored traveling salesman problem, Balanced traveling salesman problem

1 Introduction

Colored traveling salesman problem (CTSP) [1-2] is a variant of multiple traveling salesman problems (MTSP) and traveling salesman problem (TSP), which can be applied in the planning problem of multi-machine engineering systems (MES), it has a shared city set and the exclusive city sets, so that each salesman not only implements the shared task, but also performs the exclusive task. This paper provides a new variant of CTSP named CBTSP, it can be used to model the optimization problems with partially overlapped workplace. In CBTSP problem, the multiple salesmen can not only carry out the independent task in own exclusive district, but also cooperatively perform the joint task with each other in the shared district, which refers to how to cooperate for performing the multiple tasks. In the fields such as intelligent transport systems and multiple tasks cooperation, some real-world problems can be modeled by CBTSP, and the scale of generated model is usually up to large scale, thus it is necessary to study large scale CBTSP and related solving algorithms.

Because CTSP is a new problem, there are very few published papers in this field, Li *et al*. [2] firstly proposed the CTSP and used genetic algorithm (GA) for solving the problem; after that they applied three algorithms including GA with greedy algorithm (GAG), GA by hill-climbing algorithm (HCGA) and GA using simulated annealing algorithm (SAGA) to solve CTSP, but the scale of the problem is limited, where the number of city is no more than 101 [1]; Dong *et al*. [3] applied hybrid algorithms to solve the multiple balanced traveling salesmen problem, which displays that HCGA can show better performance than hybrid ITÖ algorithm, SAGA, GAG and GA. Genetic algorithm has been widely studied in recent years, the related literatures are as follows: Ardjmand *et al*. [4] applied GA to new bi-objective model; Metawa *et al*.[5] optimized bank lending decisions by using genetic algorithm based model; Ghosh *et al*. [6] used genetic algorithm by incorporating priors for medical image

Corresponding author: Xueshi Dong
E-mail: dxs_cs@163.com

segmentation; Dong *et al*. [7] proposed a hybrid algorithm based on GA for colored bottleneck TSP; the literatures [8-13] applied genetic algorithm to solve some real-world problems; Zhang *et al*. [14] utilized parallel genetic algorithm for set cover problem and large scale wireless sensor networks; Friedrich *el al*. [15] gave a compact genetic algorithm; Rashid *et al*. [16] used enhanced genetic algorithm for protein structure prediction; Lakshmi *et al*. [17] made a genetic bankrupt ratio analysis tool by using genetic algorithm.

According to the real-world applications, this paper provides a new model CBTSP, which can be applied in the optimization problems such as multiple tasks cooperation, the scheduling and deploying of the resources and goods. CBTSP is similar with CTSP, and the only difference is that they have different objective functions, therefore the nature-inspired algorithms, such as genetic algorithm, climbing hill genetic algorithm and simulated annealing genetic algorithm for CTSP, can be also used for solving CBTSP, however, while solve the problem, they are easy to fall into local optimum, which is not satisfying. In order to improve the problem, the paper proposes a new genetic algorithm called NGA to solve it, the algorithm uses dual-chromosome coding to generate the solution of problem, and crossover operator and mutation operator are used to update the solution, during the process, the crossover length can be affected by activity intensity, which is controlled by the particle radius and environment temperature, the mutation operator of NGA is same with the one of genetic algorithm. The extensive experiments show that NGA can demonstrate better performance than the compared algorithms such as SAGA, HCGA and GAG in term of solution quality.

The contributions of the paper mainly focus on the following aspects: on the one hand, this paper provides a new model called CBTSP, and extends the scale of the model to large scale in which the city number is more than 1000; on the other hand, the paper proposes a novel genetic algorithm named NGA for solving CBTSP, the experiments show the superiority of the proposed algorithm.

The other sections of the paper are as follows: the second section introduces the definition of CBTSP and relevant introduction; the third one gives the detail of NGA for solving CBTSP; the fourth one is the experiments and analysis; the last one is the conclusion and future works.

2 Colored balanced traveling salesman problem

2.1 The definition

The definition of CBTSP is similar with CTSP [1], the only difference for them is that they have different objective functions. There are m traveling salesmen and n cities for CBTSP, where $m\in Z=\{1,2,3,...\}$, $m<n$. The problem can be defined as a complete digraph G(V, E), $V=\{0,1,2,...,n-1\}$ stands for the cities set, and each edge $(i, j)\in E$, $i\neq j$, it is associated with weight w_{ij} representing the cost (e.g., distance) between the city i and city j. The cities set V is divided into m + 1 non-null sets, a set U represents the shared city set, the other set means V_i, $\forall\, i\in Z_m=\{1,2,3,...,m\}$, it shows that only a salesman i can access to it. Vertex d_i, $d_i\in(U\square V_i)$ is the depot where the salesmen i begins and ends. Color i represents that the city is visited only by salesman i. The cities set V_i (i=1, 2, 3,...,m) is colored by i, it means that only salesman i can visit it [2].

For CBTSP, there is a shared city set U. The used common one is that U can be visited by all salesmen, i.e., $\forall\, a\in U$, $c(a)=Z_m$ if $d_i=0$, $0\in U$ and $\forall\, a\in U$, $c(a)=Z_m$, the integer coding model of the corresponding CBTSP is as follows: The variable $x_{ijk}(i\neq j,\ i,\ j\in V,\ k\in Z_m)$ represents whether the k_{th} traveling salesman passes city i to j, and the variable $u_{ik}(i\in V,\ k\in Z_m)$ is the city number that the k_{th} salesman travels from depot to city i. The objective of CBTSP is to find m Hamiltonian cycles in G with the minimal difference of the maximum edge and minimum edge. The objective function of CBTSP is as follow:

$$\text{Min } f = \max(w_{ij}) - \min(w_{zk}) \ (i, j=1, 2, 3, \dots, n\text{-}1;\ z, k=1, 2, 3, \dots, n\text{-}1) \tag{1}$$

The constraint conditions of CBTSP are as follows:

$$\sum_{i=1}^{n-1} x_{0ik} = \sum_{i=1}^{n-1} x_{i0k} = 1, k \in Z_m \tag{2}$$

Formula (2) means that each salesman begins from and returns to the depot (city 0).

$$\sum_{i}\sum_{j} x_{ijk} = \sum_{i}\sum_{j} x_{jik} = 0, i \in V_k, j \in V \setminus \{U \cup V_k\}, k \in Z_m \tag{3}$$

Formula (3) represents that salesman *k* can't access other traveler's exclusive cities, furthermore, and other salesmen can't visit the k_{th}'s exclusive cities.

$$\sum_{i}\sum_{j} x_{ijl} = \sum_{i}\sum_{j} x_{jil} = 0, i \neq j, k \neq l, i \in V_k, j \in V, l \in Z_m \tag{4}$$

Formula (4) shows that traveling salesman *l*(≠*k*) can neither begin from the city of *k* exclusive nor go back to it.

$$\sum_{j=0}^{n-1}\sum_{k=1}^{m} x_{ijk} = \sum_{j=0}^{n-1}\sum_{k=1}^{m} x_{jik} = 1, j \neq i, i \in V \setminus \{0\} \tag{5}$$

The formula (5) demonstrates each city except the depot 0 must be visited by a salesman exactly once.

$$\sum_{l} x_{jlk} = \sum_{i} x_{ijk}, i \neq j \neq l, j \in U, i, l \in V_k \cup U \tag{6}$$

The formula (6) means each salesman must access and withdraw from a shared city at the same time.

2.2 CBTSP and CTSP

CTSP and CBTSP are both a version of MTSP and TSP. CBTSP and CTSP not only have shared city set, but also occupy exclusive city set, the shared cities can be visited by all salesmen, but exclusive cities are only accessed by the appointed salesman, other salesmen have no authority to visit them. CBTSP and CTSP have different objective functions: the objective of CBTSP is to find the tours where the difference of the maximum edge and minimum edge is minimized, and the objective function of CTSP is to search the tours in which the total traveling distance is as small as possible [1].

2.3 CBTSP related theory

CBTSP is NP-hard problem. CTSP is a variant of MTSP and TSP, under some condition, CTSP can be transformed into MTSP or TSP, it has been proved that CTSP is a NP-hard problem [2]. With changing the objective function of CTSP, it can be transformed into CBTSP, the time complexity of this model will not change due to this operation, thus CBTSP is also NP-hard problem.

2.4 CBTSP applications

The multiple balanced traveling salesmen problem [3] can't be used to model the optimization problems with cooperative task. However, CBTSP can be applied in the optimization and planning problems of persons and vehicles with cooperative and exclusive tasks such as the multiple tasks cooperation, the planning and deploying of the resource and goods. For example, there are six people who uniformly deploy goods, while the goods are too many in a district, the six persons need cooperate to allocate the goods, when the goods are few, each of them will independently carries out the task in six different districts, the objective is to find six tours in which the goods are uniformly deployed. The basic elements of such a problem, i.e., objective, persons, and tasks, can respectively match the objective, salesmen, and cities of CBTSP, thus it can be modeled by CBTSP. It is not limited to the given instance, and the model can be applied to the kinds of optimization problems with independent tasks and cooperative task.

3 NGA for CBTSP

3.1 Solution representation

The literature [1] uses dual-chromosome coding to represent the solution of CTSP. This paper also utilizes the method to code the solution of CBTSP. As shown in figure 1, the city chromosome is the permutation of the *n*-1 cities, while the salesman chromosome is the permutation of the salesmen corresponding to the cities.

We give an example to show the coding, in figure 1, for example, there is a CBTSP problem with 9 cities and 2 salesmen, city 1 to city 3 are the exclusive cities of salesman 1, city 4 to city 6 belong to the exclusive cities of salesman 2, city 7 to city 9 are the shared cities of the two salesmen. If the starting and ending point (depot 0) is contained, the visiting path of salesman 1 is 0-9-3-1-7-2-0, and the access route of salesman 2 is 0-4-5-6-8-0.

City chromosome:

9	4	3	5	6	1	7	8	2

Salesman chromosome:

1	2	1	2	2	1	1	2	1

Fig.1. an example of dual-chromosome coding for CBTSP

3.2 The steps of NGA

The proposed algorithm is a novel genetic algorithm based on ITÖ process [18-19], the solutions of problem are considered to be particles, and the activity intensity of particles is affected by particles radius and environment temperature. The crossover length of crossover operator is controlled by the activity intensity, the larger the intensity is, the longer the length becomes. During the process of solving, the *i*th particle is crossed with the best solution.

The steps of NGA for CBTSP are shown in algorithm 1:

Algorithm 1 *Best solution*←NGA for CBTSP

```
1:  Set parameters of the algorithm;
2:  Initialize $M$ particles and initial temperature $T$;
3:  CBTSP ←read dataset;
4:  Calculate the fitness of all particles, and record best one;
5:  Compute the radius by formula (8);
6:  Compute environment temperature by formula (9);
7:  Compute activity intensity by formula (10);
8:  Sort all the particles by their fitness;
9:  Calculate crossover length by formula (11);
10: while stopping condition is not satisfied do
11:         for particle s in all particles do
12:             Perform the crossover operator and mutation operator to
                generate a new solution;
13:         end for
14:    Calculate the fitness and memorize the best;
15:    Sort all the particles by their fitness;
16:    Update annealing temperature;
17:    Update crossover length of all particles;
18: end while
19: return Best solution;
```

In algorithm 1, step 1 is the parameters initialization, step 2 is to initialize the population *M* and initial temperature *T*, step 3 is to read the CBTSP data, step 4 is to compute the fitness of the particles, and save the best one, steps 4 to 6 calculate particle radius, environment temperature and activity intensity, step 8 is to classify the particles according to fitness, step 9 computes the crossover length by formula (11), steps 10 to 13 carry out crossover operator and mutation operator, the former is the key operator of NGA based on ITÖ process, which can randomly select some cities of city chromosome, then they are crossed with the corresponding ones of best solution, the crossover operator corresponds to the selection operator and crossover operator of genetic algorithm, the mutation operator is the same with the mutation operator of genetic algorithm, steps 16 to 17 update environment temperature and crossover length.

3.3 NGA algorithm

NGA algorithm includes five parts: particle radius, environment temperature, activity intensity, crossover operator and mutation operator. Among them, particle radius and environment temperature are influence factors, which can affect the activity intensity, it can control the crossover length of the crossover operator [7]. For NGA, the crossover operator is redesigned based on ITÖ process, which corresponds to the selection operator and crossover operator of genetic algorithm; mutation operator is same with the mutation operator of genetic algorithm [1]. The details of particle radius, environment temperature, activity intensity, crossover operator for the proposed algorithm are as follows:

(1) Particle radius

The formula can be defined as follows:

$$r(x) = g(f(x)) \tag{7}$$

where x stands for the particle of current swarm, $f(x)$ represents the fitness, $g(x)$ is monotonic function. The paper computes the particle radius according to classification, N particles are classified by the fitness based on the best to worst order, which are represented by x_1, x_2,…., x_N, a version of particle radius is computed by:

$$r(x_i) = \frac{(N-i)(r_{max} - r_{min})}{N-1} + r_{min} \tag{8}$$

where r_{max} and r_{min} respectively represent the maximum and minimum particle radius, all particle radii are uniformly distributed in r_{max} and r_{min}, by default, r_{max} is set as 1, r_{min} is 0.

(2) Environment temperature

For simulated annealing algorithm, during the process of iteration, the environment temperature is gradually reduced, it is defined by the below formula:

$$T_i = \rho \cdot T_{i-1} \tag{9}$$

where T_i represents the temperature at the ith scheduling time, ρ stands for the annealing coefficient, which can control the speed of the temperature dropping, by default, it is set as 0.9.

(3) Activity intensity

Activity intensity controls the movement intensity of particles, the activity intensity I of current particle x_i is computed by the below formula:

$$I_i = \frac{(e^{-\lambda r_i} - e^{-\lambda r_{max}})}{(e^{-\lambda r_{min}} - e^{-\lambda r_{max}})} \cdot e^{-\frac{1}{T}} \tag{10}$$

where I represents the activity intensity, r_i is the radius of particle x_i, T stands for temperature.

(4) Crossover operator

Because of crossover operator, particles can change in the solution space, under the environment influence, the particles (solutions) are crossed with the best solution, which can generate a new solution, it can display strong global search ability, and keep diversity of solution. The crossover operator contains two parts including crossover length and crossover process, the crossover length is controlled by the activity intensity, which is computed by environment temperature and particle radius [7]. The crossover length is calculated by the below formula:

$$L_i = \gamma \cdot I_i \cdot l \tag{11}$$

where L_i represents the crossover length of the ith particle, γ is random obeying uniform distribution number, it is between 0 and 1, l stands for the length of dual-chromosome coding.

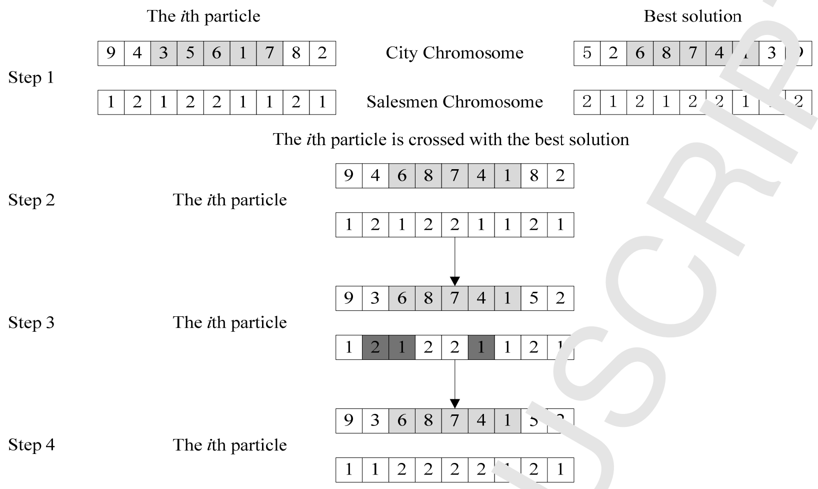

Fig.2. an example of crossover operator

As shown in figure 2, the crossover process of algorithm NGA is as follows:

Step 1: A starting position in [1, l-L_i] in city chromosome is randomly selected, and the continuous L_i positions in the gray segment are crossed with the best solution.

Step 2: Replace the cities in gray segment by the corresponding cities of best solution. For example, the swap relationship is: 3-6, 5-8, 6-7, 1-4, 7-1.

Step 3: According to the swap relationship, replace the redundant cities outside the gray segment in the ith particle until there are no redundant cities.

For example, in the step 2, there is number 4 outside the gray segment, and number 4 is also in the gray segment, which is called redundancy. Therefore, the 4 outside the gray segment is needed to be replaced. According to the swap relationship, the 4 in best solution corresponds to the 1 in ith particle, thus 4 is replaced by 1, but 1 is still redundant. Based on the rule, 1 is replaced by 7, it still doesn't meet the condition, then 7 is replaced by 6, which is also redundant, thus 6 is replaced by 3, which can meet the condition. Finally, the final value is 3.

Step 4: Correct the wrong assignment value in salesman chromosome. For example, in the third step, the city chromosomes 3, 6 and 4 correspond the wrong salesmen, therefore it needs correction, and the revised result is shown in step 4.

After the four steps, the particle (solution) finishes the crossover operator, and carries out the updating of the solution for solving CBTSP problem.

4 Experiments and Analysis

4.1 The small and medium scale CBTSP

We make experiments to analyze the performance of the different algorithms for CBTSP. The computer environment is as follow: Intel® Core™ i7-6700 running Windows 7 with 3.40 GHz processor and 8.00 GB RAM. The experiments are developed based on Java.

The initial parameters of GAs are from the CTSP paper [1]: the population size is 30, crossover probability as 0.7, and mutation probability is 0.1. The parameters of the SAGA are as follows: the initial temperature as 100, the total time of cooling is 60, the step length at each temperature is to be 30, and annealing coefficient as 0.9. The parameters of NGA: the initial population is 150, and the initial environment temperature is 1000. All the algorithms have the same stopping condition. Each algorithm runs ten times. The below table is the small and medium scale experiment data.

Table 1 the small and medium scale experiments data for CBTSP

Instance scale	**City count**	**Salesman count**	**Shared city**	**Exclusive city**
Small	***n***	***m***	***s***	***e***
1	21	2	11	5
2	21	3	9	4
3	31	2	19	6
4	31	3	16	5
5	31	4	15	4
6	41	2	21	10
7	41	3	23	6
8	41	4	21	5
9	51	3	21	10
10	51	4	21	7, 8
11	51	5	21	6
12	76	3	31	15
13	76	4	36	10
14	76	5	26	10
15	76	6	40	6
Medium				
16	101	4	21	20
17	101	5	53	10
18	101	6	41	10
19	101	7	21	10
20	202	12	62	10
21	202	25	77	5
22	202	35	62	4
23	431	12	191	20
24	431	25	181	10
25	431	40	231	5
26	655	17	145	30
27	655	25	155	20
28	655	35	160	15

In the table 1, *n* is the number of city, *m* is the number of salesmen, *s* is the city number of shared set of CBTSP, and *e* is the city number of an exclusive data. The data with the city number during 21 to101 is provided in the CTSP paper [1], and the data with city number from 202 to 655 is made by the original TSP data (The TSPLIB Symmetric Traveling Salesman Problem Instances) published on web.

The following figures are the solving routes figure of the four algorithms for CBTSP.

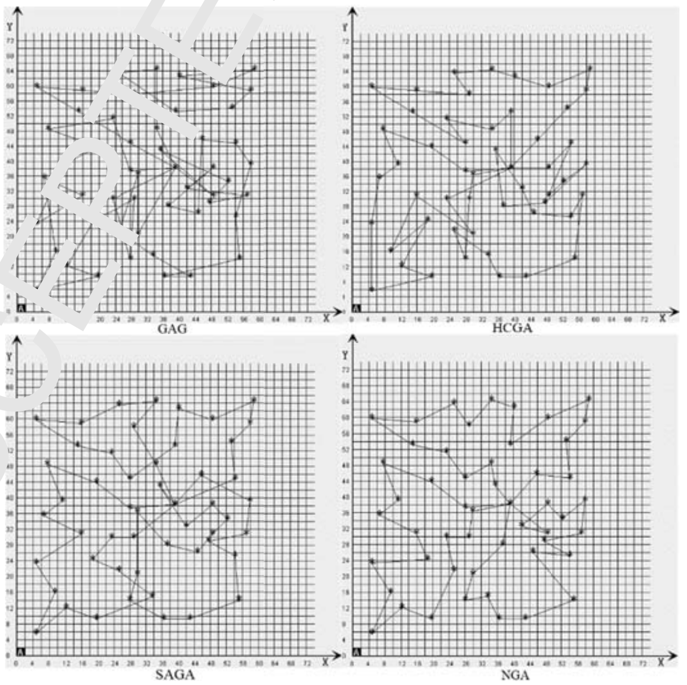

Fig.3. the optimal routes of the algorithms for CBTSP with *n*=51 and *m*=3 (Unit: km)

In the figure 3, it is the best solution case of the four algorithms for CBTSP using n=51 and m=3, the upper-left small figure is GAG for CBTSP with n=51 and m=3, the upper-right one is HCGA for CBTSP, the bottom-left one means SAGA for the problem, the bottom-right one represents NGA for solving the CBTSP problem. The detail of the four algorithms for CBTSP is as follows: GAG: average solution 17.7, iteration 68322; HCGA: average solution 17.3, iteration 37604; SAGA: average solution 15.2, iteration 2168; NGA: average solution 15.8, iteration 9334.

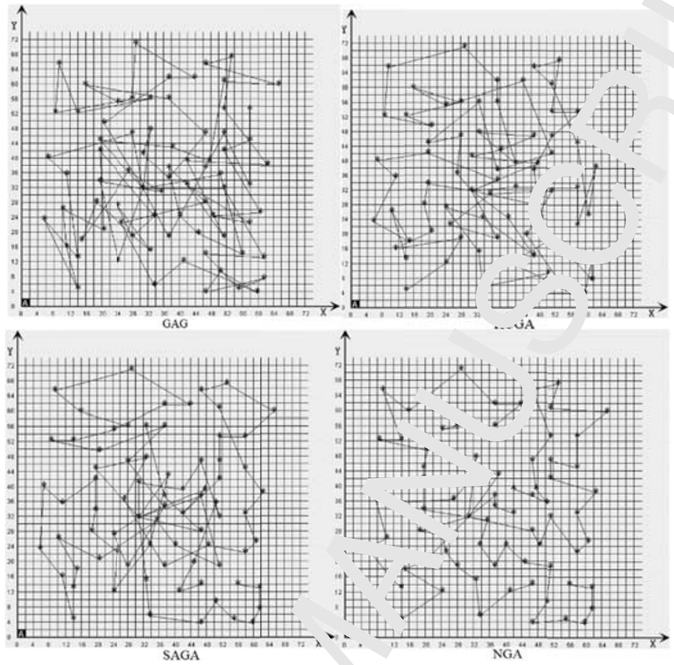

Fig.4. the optimal routes of the algorithms for CBTSP with n=76 and m=4 (Unit: km)

Figure 4 is the best solution case of the four algorithms for CBTSP by n=76 and m=4, the top-left small figure is GAG for the problem, the top-left one represents HCGA for CBTSP, the lower left one is SAGA for CBTSP, the lower right one is NGA for solving the problem. The details are as follows: GAG: average solution 20.7, iteration 2247[illegible], HCGA: average solution 20.0, iteration 16934; SAGA: average solution 19.6, iteration 3[illegible]02; NGA: average solution 16.9, iteration 5556.

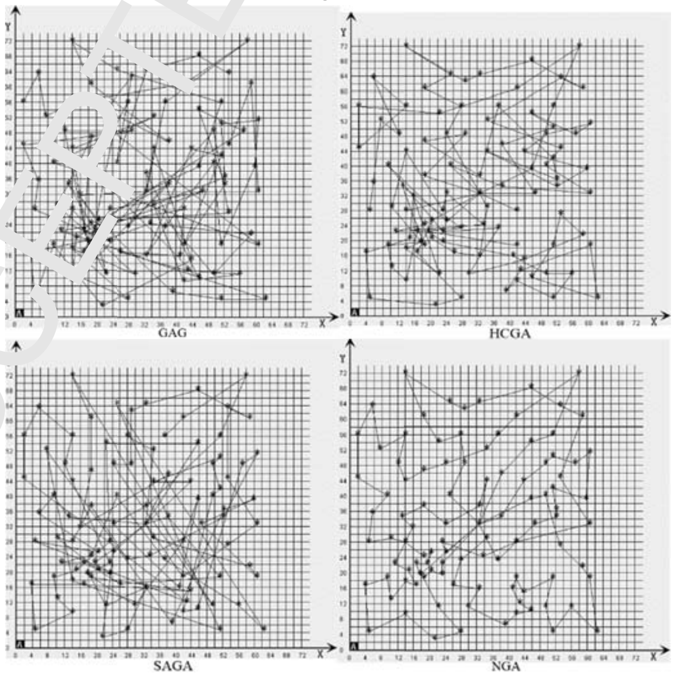

Fig.5. the optimal routes of the algorithms for CBTSP with n=101 and m=5 (Unit: km)

Figure 5 is the solving routes figures of the four algorithms for CBTSP by n=101 and m=5, the top-left small figure stands for GAG for the problem, the top-left one represents HCGA for CBTSP, the lower left one is SAGA for solving the problem, the lower right one is NGA for CBTSP. The details of the four algorithms are as follows: GAG: average solution 33.7, iteration 19674; HCGA: average solution 29.1, iteration 17134; SAGA: average solution 29.5, iteration 1072; NGA: average solution 22.1, iteration 5049.

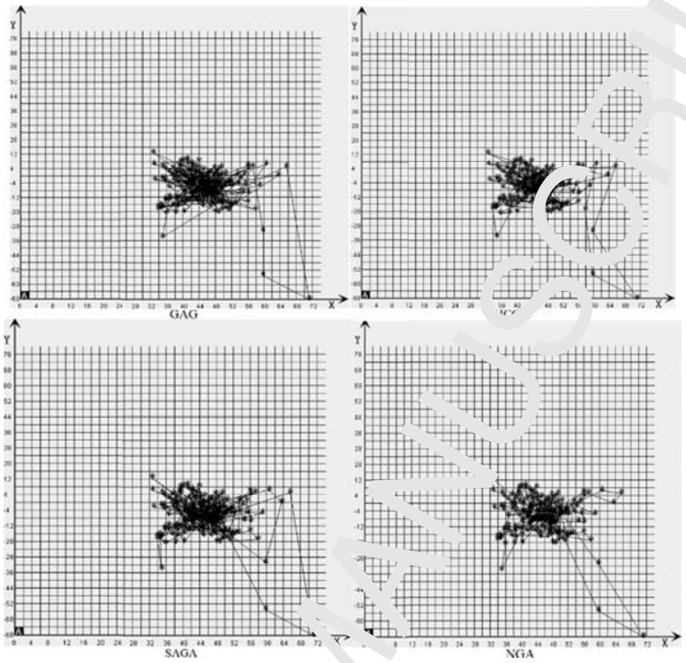

Fig.6. the optimal routes of the algorithms for CBTSP with n=202 and m=12 (Unit: km)

Figure 6 is the solution case of the four algorithms for CBTSP by n=202 and m=12, the top-left small figure stands for GAG for the problem, the top-left one represents HCGA for CBTSP, the lower left one is SAGA for the problem, the lower right one is NGA.

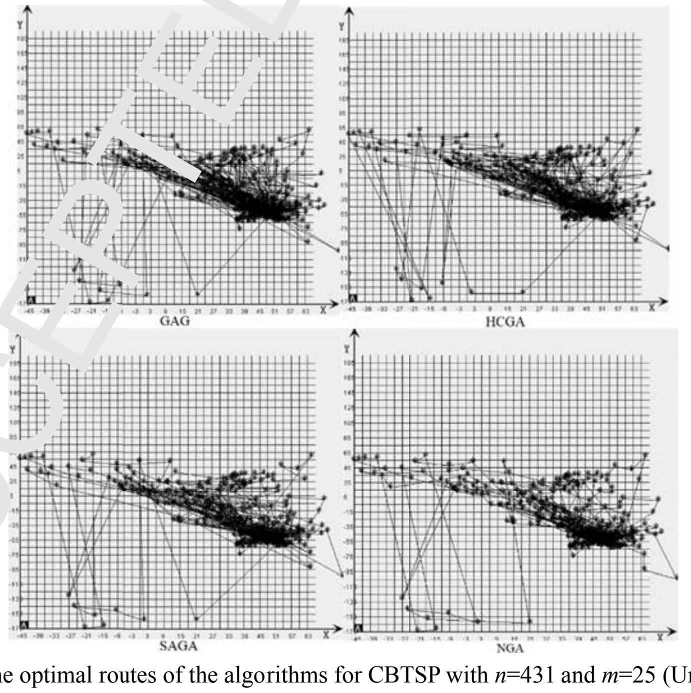

Fig.7. the optimal routes of the algorithms for CBTSP with n=431 and m=25 (Unit: km)

Figure 7 is the solving routes of the four algorithms to solve CBTSP by n=431 and m=25, the top left figure represents the GAG for solving the problem, the upper right small figure means HCGA for

CBTSP, the bottom left one is SAGA for the problem, the lower right one is NGA for CBTSP. The details are as follows: GAG: average solution 13133.9, iteration 3183; HCGA: average solution 12487.8, iteration 2885; SAGA: average solution 12786.7, iteration 1265; NGA: average solution 12967.0, iteration 859.

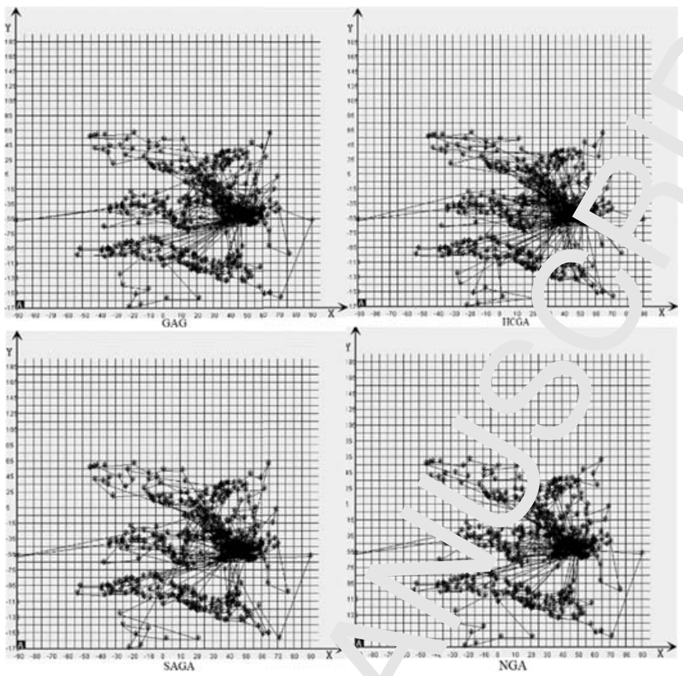

Fig.8. the optimal routes of the algorithms for CBTSP with n=655 and m=33 (Unit: km)

The below table 2 is the experiments of the five algorithms for small scale and medium scale CBTSP by the same stopping condition.

Table 2 the experiment results of the five algorithms for CBTSP by running 20s as the stopping condition (Unit: mean km)

Instance	***n***	***m***	**GA**		**GAG**		**HCGA**		**SAGA**		**NGA**	
			Mean	Iteration	Mean	Iteration	Mean	Iteration	Mean	Iteration	Mean	Iteration
Small												
1	21	2	6.8	75263	6.5	103408	6.5	22320	6.2	2982	8.4	26907
2	21	3	7.6	49756	6.[illegible]	23344	7.4	29057	6.8	6547	9.4	21610
3	31	2	10.4	3683[illegible]	9.7	103932	9.5	24244	8.4	2159	11.7	15727
4	31	3	11.9	54[illegible]17	10.2	63327	9.6	62433	10.0	1976	13.9	15036
5	31	4	9.3	65924	9.[illegible]	40510	8.2	28725	8.0	4228	13.5	14982
6	41	2	13.6	[illegible]	13.3	28745	12.6	33010	10.4	1533	11.6	11504
7	41	3	14.0	5141[illegible]	14.0	54774	13.8	64254	12.5	4628	13.5	11337
8	41	4	14.5	36[illegible]93	12.7	45544	12.7	48058	12.4	2869	15.9	11188
9	51	3	18.6	38[illegible]	17.7	68322	17.3	37604	15.2	2168	15.8	9334
10	51	4	16.2	41010	18.8	33841	16.3	46805	12.7	5501	17.3	8932
11	51	5	18.[illegible]	3307[illegible]	18.5	28155	15.3	22417	15.3	1953	19.2	8679
12	76	3	24.8	22[illegible]94	23.0	28589	20.0	23793	18.8	3189	15.7	5922
13	76	4	[illegible].4	[illegible]359	20.7	22477	20.0	16934	19.6	3702	16.9	5556
14	76	5	22.1	18959	21.3	29226	19.5	17376	17.9	2595	18.5	5723
15	76	6	27.4	14967	28.0	9769	20.9	20340	22.9	2290	21.0	5538
Medium												
16	101	4	29.1	18094	26.4	22180	25.8	17724	21.5	1474	18.8	5485
17	101	5	3[illegible].4	7846	33.7	19674	29.1	17134	29.5	1072	22.1	5049
18	101	6	29.[illegible]	19949	24.2	39935	22.6	18430	25.6	1845	20.5	5101
19	101	7	[illegible].4	18971	25.3	17034	21.9	17154	24.0	2789	22.5	5409
20	[illegible]	12	2339.2	12722	2257.4	11883	2267.2	10962	2425.8	2761	2626.6	2388
21	202	25	2393.6	5338	2335.3	7271	2416.2	10050	2490.0	2093	2626.0	2003
22	202	35	2488.2	5244	2498.7	3538	2506.8	6425	2535.6	2143	2647.0	1755
23	431	12	10767.4	5627	9607.0	5478	8662.3	3924	8017.4	1315	7426.9	915
24	431	25	13143.8	2938	13133.9	3183	12487.8	2885	12786.7	1265	12967.0	859
25	431	40	14736.3	2931	14732.0	3163	13862.2	2695	14749.2	1103	10962.3	695
26	655	17	11349.4	3741	12682.7	2412	9914.7	2528	10694.8	868	8873.1	582
27	655	25	12561.9	2811	12091.0	2559	10457.4	2360	9827.2	857	10470.9	539
28	655	33	12507.0	2353	13115.8	1725	11435.4	1537	11488.9	827	11250.6	487

Figure 8 is the best routes of the four algorithms for CBTSP by n=655 and m=33, the top left figure is the GAG for solving the problem, the upper right one means HCGA for CBTSP, the bottom left one represents SAGA for the problem, the lower right one stands for NGA for CBTSP. The detail is as follows: GAG: average solution 13115.8, iteration 1725; HCGA: average solution 11435.4, iteration 1537; SAGA: average solution 11488.9, iteration 827; NGA: average solution 11250.6, iteration 487.

In the table 2, it is the experiment results of five algorithms to solve the small and medium scale CBTSP, the mean is the average solution quality of each algorithm running 10 times for the problem, the iteration represents the average iteration times of algorithm running 10 times for obtaining the best solution. Among them, GA, GAG, HCGA and SAGA are from the CTSP literature [1]. NGA is valid swarm intelligence algorithm for the optimization problems. By the experiments, it shows that NGA is effective to solve the small scale and medium scale problem.

4.2 The large scale CBTSP

The computer environment and the parameters setting of all algorithms are the same with the one of the experiments for small scale and medium scale CBTSP problem. The stopping conditions such as fitness function evaluation and maximum number of iterations usually have problems, which are not fair stopping conditions [20-21]. In order to make up the flaw of single stopping condition, we use two stopping conditions to make experiments for large scale CBTSP, each algorithm runs same time for solving the problem, the first one is running 60s for each algorithm, the second one runs 180s. The following table 3 is the large scale data of CBTSP.

Table 3 the large scale experiments data for CBTSP

Instance scale	**City count**	**Salesman count**	**Shared city**	**Exclusive city**
Large	*n*	*m*	*s*	*e*
1	2461	3	661	600
2	2461	6	661	300
3	2461	12	661	150
4	2461	24	661	75
5	2461	30	661	60
6	3461	3	461	1000
7	3461	6	461	500
8	3461	12	461	250
9	3461	24	461	125
10	3461	30	461	100
11	3461	40	461	75
12	5397	20	397	250
13	5397	30	897	150
14	5397	40	1397	100
15	5397	50	397	100
16	5397	60	597	80
17	7397	20	1397	300
18	7397	30	1397	200
19	7397	40	1397	150
20	7397	50	1397	120
21	7397	60	1397	100

In table 3, there are 21 instances, *n* is from 2461 to 7397, *m* is from 3 to 60, the large scale data is made according to the original TSP (The TSPLIB Symmetric Traveling Salesman Problem Instances) data which is published on web.

The following tables are the experiments of the algorithms for CBTSP by running 60s as the stopping condition.

The compared algorithms GA, GAG and HCGA are from the literatures [1-2], the modified genetic algorithm SAGA is from the literature [1], the hill-climbing algorithm are used to optimize genetic algorithm two times (the algorithm is named HHGA) [22], the simulated annealing algorithm is used for optimization after the operators of genetic algorithm (the algorithm is called GASA) [23], hill-climbing as the local search method (neighborhood search) is used after the crossover and mutation of genetic algorithm (the algorithm is GAHC) [24], the modified genetic algorithm GAQSA refers to the literature [25], NGA is the proposed algorithm in this paper.

Table 4 the solution quality of the five algorithms for large scale CBTSP with running 60s as the stopping condition (Unit: km)

Instance	***n***	***m***	**GA**		**GAG**		**HCGA**		**SAGA**		**NGA**	
Large			Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean
1	2461	3	3939.0	3985.1	1322.0	1721.6	1103.0	1628.4	1324.0	[illegible]	**786.0**	**846.4**
2	2461	6	3951.0	3991.4	1915.0	2201.9	2064.0	2481.1	1536.0	[illegible]	**1322.0**	**1439.0**
3	2461	12	3923.0	4014.1	2831.0	3116.6	2892.0	3153.9	2078.0	[illegible]	**1906.0**	**2122.7**
4	2461	24	3986.0	4040.5	3060.0	3382.1	3204.0	3432.3	**1747.0**	[illegible]	2479.0	**2667.1**
5	2461	30	3938.0	4049.1	2930.0	3267.0	2992.0	3246.2	[illegible]	2810.2	2576.0	**2753.3**
6	3461	3	4185.0	4276.9	1364.0	2040.9	1633.0	2012.5	1504.0	1917.2	**1235.0**	**1278.3**
7	3461	6	4205.0	4276.4	1968.0	2384.7	2040.0	2380.8	1770.0	2091.0	**1636.0**	**1859.0**
8	3461	12	4232.0	4273.4	2919.0	3201.0	2793.0	3131.1	2283.0	2762.1	**2061.0**	**2366.2**
9	3461	24	4189.0	4311.4	3603.0	3726.8	3603.0	3722.6	3040.0	3447.8	**2761.0**	**2960.7**
10	3461	30	4183.0	4283.6	3237.0	3592.7	3337.0	3612.4	3034.0	3489.2	**2911.0**	**3170.3**
11	3461	40	4196.0	4288.7	3456.0	3800.1	3667.0	[illegible]	[illegible]	3561.3	3141.0	**3280.3**
12	5397	20	735474.0	745851.4	702354.0	727316.6	707444.0	[illegible]	674393.0	703255.7	**624264.0**	**652183.1**
13	5397	30	750620.0	757848.6	731087.0	750034.8	726710.0	[illegible]	714826.0	739228.3	**676234.0**	**698300.6**
14	5397	40	431785.0	435895.5	379393.0	385715.3	378446.0	386210.2	378006.0	381448.9	**211532.0**	**261310.7**
15	5397	50	435987.0	442208.9	379333.0	391584.9	380971.0	[illegible]	378198.0	380183.5	**274364.0**	**310120.5**
16	5397	60	435639.0	441107.6	391221.0	410408.0	380520.0	397614.2	379429.0	390184.4	**275904.0**	**330702.3**
17	7397	20	698756.0	706637.9	586709.0	630030.6	594284.0	[illegible]	594832.0	633248.2	**564760.0**	**585141.5**
18	7397	30	727724.0	742003.4	**620816.0**	652410.4	618871.0	651421.2	632820.0	660184.6	588471.0	**604897.7**
19	7397	40	462935.0	466051.1	380267.0	387829.2	378808.0	387217.4	379482.0	387663.3	**263991.0**	**277376.3**
20	7397	50	462308.0	466342.1	383740.0	388956.6	[illegible]	389618.5	378016.0	387109.5	**286046.0**	**306737.8**
21	7397	60	462830.0	465983.5	384435.0	390511.0	386792.0	389408.1	383588.0	388659.3	**280287.0**	**304233.9**

In table 4, *n* is the city number, *m* stands for the number of salesmen, best and mean represent the best solution and average solution of the algorithms running 10 times for CBTSP. The table shows that NGA can show better solution quality than the compared algorithms.

Table 5 the solution quality of the five algorithms for large scale CBTSP with running 60s as the stopping condition (Unit: km)

Instance	***n***	***m***	**HHGA**		**GASA**		**GAHC**		**GAQSA**		**NGA**	
Large			Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean
1	2461	3	1412.0	1746.9	1230	1489.2	1434.0	1756.6	1333.0	1620.5	**786.0**	**846.4**
2	2461	6	1689.0	2341.1	[illegible]	1894.7	1834.0	2391.0	1901.0	2174.7	**1322.0**	**1439.0**
3	2461	12	2447.0	2863.7	[illegible]	2686.0	2696.0	3077.0	2189.0	2460.7	1906.0	**2122.7**
4	2461	24	3356.0	3492.6	**2015.0**	2878.7	3149.0	3363.4	2588.0	2691.7	2479.0	**2667.1**
5	2461	30	2774.0	[illegible]	2899.0	3132.7	2741.0	3234.7	2607.0	**2709.0**	**2576.0**	2753.3
6	3461	3	1654.0	2051.0	[illegible]	1985.5	1397.0	1769.0	1355.0	1877.4	**1235.0**	**1278.3**
7	3461	6	2173.0	[illegible]	2122.0	2375.2	2015.0	2332.1	1973.0	2243.3	**1636.0**	**1859.0**
8	3461	12	2924.0	[illegible]	2726.0	2989.3	2848.0	3161.6	2506.0	2796.9	**2061.0**	**2366.2**
9	3461	24	3407.0	[illegible]	3011.0	3424.3	3524.0	3676.5	3030.0	3262.8	**2761.0**	**2960.7**
10	3461	30	3252.0	[illegible]	2941.0	3437.8	3288.0	3602.4	3050.0	3416.0	**2911.0**	**3170.3**
11	3461	40	3458.0	3736.6	**2962.0**	3480.6	3468.0	3710.0	3125.0	3435.8	3141.0	**3280.3**
12	5397	20	711914.0	[illegible]	680495.0	704133.7	695960.0	725712.0	693410.0	701328.9	**624264.0**	**652183.1**
13	5397	30	[illegible]	[illegible]	710104.0	731490.9	719362.0	744071.7	685882.0	732548.3	**676234.0**	**698300.6**
14	5397	40	377224.0	380031.2	379412.0	380972.9	378961.0	386469.8	377605.0	379197.5	**211532.0**	**261310.7**
15	5397	50	378413.0	390743.5	377818.0	384265.8	379408.0	386357.6	378394.0	380634.0	**274364.0**	**310120.5**
16	5397	60	[illegible]	400120.4	379414.0	383077.8	379474.0	397055.9	379325.0	387178.0	**275904.0**	**330702.3**
17	7397	20	[illegible]	617729.3	599513.0	620764.0	592693.0	624653.8	589409.0	628891.6	**564760.0**	**585141.5**
18	7397	30	[illegible]	664049.8	636528.0	649394.3	627827.0	649173.3	624777.0	646689.1	**588471.0**	**604897.7**
19	7397	40	[illegible]	387417.0	384432.0	388763.3	383588.0	388002.8	379482.0	385691.0	**263991.0**	**277376.3**
20	7397	50	383740.0	387242.9	379693.0	388556.9	381267.0	389270.0	381267.0	385665.8	**286046.0**	**306737.8**
21	7397	60	383743.0	388602.4	383585.0	387802.5	384151.0	388970.2	385559.0	388533.5	**280287.0**	**304233.9**

From table 5, it shows that NGA can display better solution quality than other algorithms. The following figure 9 and figure 10 are average solution quality of the algorithms for CBTSP.

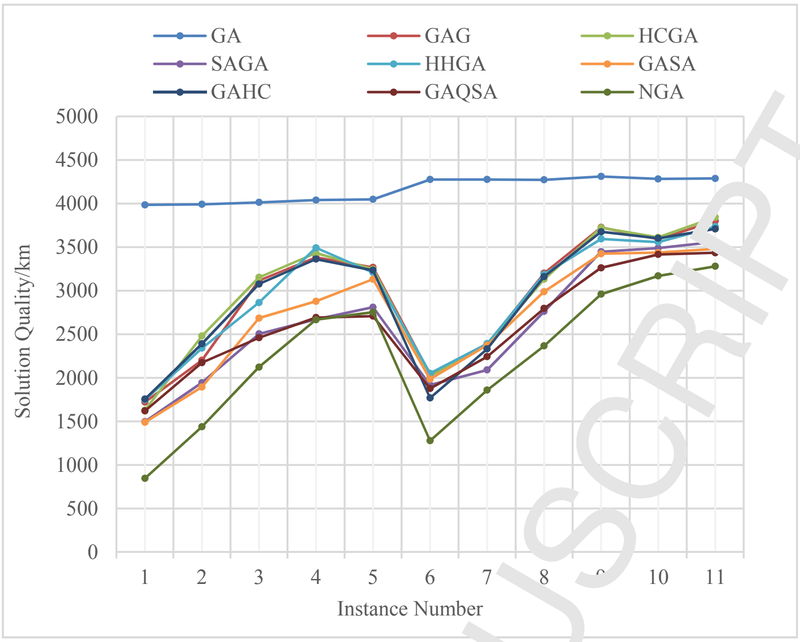

Fig.9. the average solution quality of the algorithms for large scale CBTSP (Unit: km)

In figure 9 and figure 10, the lateral axis stands for the order number of the instance. For example, the number 2 corresponds to the instance with n=2461 and m=6; vertical axis represents the mean solution quality of the algorithms for the problem.

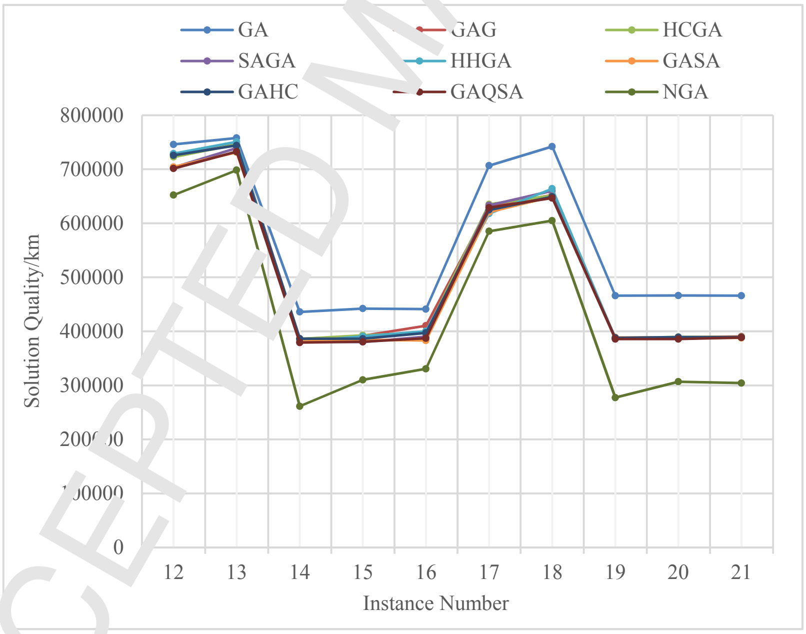

Fig.10. the average solution quality of the algorithms for large scale CBTSP (Unit: km)

Figure 9 and figure 10 show that NGA demonstrates better mean solution quality than GAQSA, GAHC, GASA, HHGA, SAGA, HCGA, GAG and GA.

In order to test the effectiveness of the algorithms for the problem, we use the best solution deviation PD_{best} and the average best solution deviation PD_{av} to make significance test [26]. The percentage deviation is computed based on the data of table 4 and table 5, and the computation formula and method are given in the literature [26]. The computation results are shown in table 6 and table 7.

Table 6 the percentage deviation of the five algorithms for large scale CBTSP with running 60s as the stopping condition

Instance	***n***	***m***	**GA**		**GAG**		**HCGA**		**SAGA**		**NGA**	
Large scale			PD_{best}	PD_{av}	PD_{best}	PD_{av}	PD_{best}	PD_{av}	PD_{best}	PD_{av}	PD_{best}	PD_{av}
1	2461	3	401.1	370.8	68.1	103.4	40.3	92.3	68.4	76.[illegible]	**0.0**	**0.0**
2	2461	6	198.8	177.3	44.8	53.0	56.1	72.4	16.1	35.1	**0.0**	**0.0**
3	2461	12	122.8	89.1	60.8	46.8	64.3	48.5	18.0	[illegible]7.9	8.2	**0.0**
4	2461	24	128.1	51.4	75.1	26.8	83.4	28.6	**0.0**	0.1	41.9	**0.0**
5	2461	30	66.0	47.0	23.5	18.6	26.1	17.9	**0.0**	2.0	8.6	**0.0**
6	3461	3	238.8	234.5	10.4	59.6	32.2	57.4	21.7	[illegible]	**0.0**	**0.0**
7	3461	6	157.0	130.0	20.2	28.2	24.6	28.0	8.1	12.4	**0.0**	**0.0**
8	3461	12	105.3	80.6	41.6	35.2	35.5	32.3	10.7	16.7	**0.0**	**0.0**
9	3461	24	51.7	45.6	30.4	25.8	30.4	25.7	10.1	16.4	**0.0**	**0.0**
10	3461	30	61.1	35.1	24.6	13.3	28.5	13.9	1[illegible]	10.0	12.1	**0.0**
11	3461	40	60.1	30.7	31.9	15.8	39.9	17.1	**0.0**	8.5	19.8	**0.0**
12	5397	20	17.8	14.3	12.5	11.5	13.3	10.7	8.0	7.8	**0.0**	**0.0**
13	5397	30	11.0	8.5	8.1	7.4	7.4	6.8	5.7	5.8	**0.0**	**0.0**
14	5397	40	104.1	66.8	79.3	47.6	78.9	47.7	[illegible]	45.9	**0.0**	**0.0**
15	5397	50	58.9	42.5	38.2	26.2	38.8	26.5	37.8	22.5	**0.0**	**0.0**
16	5397	60	57.8	33.3	41.7	24.1	37.9	[illegible].2	37.5	17.9	**0.0**	**0.0**
17	7397	20	23.7	20.7	3.8	7.6	5.2	8.5	5.3	8.2	**0.0**	**0.0**
18	7397	30	23.6	22.6	5.4	7.8	5.1	7.6	7.5	9.1	**0.0**	**0.0**
19	7397	40	75.3	68.0	44.0	39.8	43.4	3[illegible].6	43.7	39.7	**0.0**	**0.0**
20	7397	50	61.6	52.0	34.1	26.8	34.7	[illegible].0	32.1	26.2	**0.0**	**0.0**
21	7397	60	65.1	53.1	37.1	28.3	37.[illegible]	27.9	36.8	27.7	**0.0**	**0.0**
	Average		99.5	79.7	35.0	31.1	36.4	31.3	22.0	21.7	**4.3**	**0.0**

Table 7 the percentage deviation of the five algorithms for large scale CBTSP with running 60s as the stopping condition

Instance	***n***	***m***	**HHGA**		**GASA**		**GAHC**		**GAQSA**		**NGA**	
Large scale			PD_{best}	PD_{av}	PD_{best}	PD_{av}	PD_{best}	PD_{av}	PD_{best}	PD_{av}	PD_{best}	PD_{av}
1	2461	3	79.6	106.3	56.4	75.9	82.4	107.5	69.5	91.4	**0.0**	**0.0**
2	2461	6	27.7	62.6	14.2	31.6	38.7	66.1	43.7	51.1	**0.0**	**0.0**
3	2461	12	39.5	34.9	**0.0**	26.5	53.7	44.9	24.8	15.9	8.6	**0.0**
4	2461	24	66.5	30.9	**0.0**	7.9	56.2	26.1	28.4	0.9	23.0	**0.0**
5	2461	30	7.6	18.4	12.5	15.6	6.4	19.4	1.2	**0.0**	**0.0**	1.6
6	3461	3	33.9	60.4	25.8	55.3	13.1	38.3	9.7	46.8	**0.0**	**0.0**
7	3461	6	32.8	28.6	29.[illegible]	27.7	23.1	25.4	20.5	20.6	**0.0**	**0.0**
8	3461	12	41.8	34.[illegible]	32.2	26.3	38.1	33.6	21.5	18.2	**0.0**	**0.0**
9	3461	24	23.3	21.[illegible]	9.0	15.6	27.6	24.1	9.7	10.2	**0.0**	**0.0**
10	3461	30	11.7	12.2	1[illegible]	8.4	12.9	13.6	4.7	7.7	**0.0**	**0.0**
11	3461	40	16.7	13.9	**0.0**	6.1	17.0	13.0	5.5	4.7	6.0	**0.0**
12	5397	20	14.0	11.[illegible]	9.0	7.9	11.4	11.2	11.0	7.5	**0.0**	**0.0**
13	5397	30	8.3	7.5	5.0	4.7	6.3	6.5	1.4	4.9	**0.0**	**0.0**
14	5397	40	78.[illegible]	45.4	79.3	45.7	79.1	47.8	78.5	45.1	**0.0**	**0.0**
15	5397	50	37.[illegible]	25.9	37.7	23.9	38.2	24.5	37.9	22.7	**0.0**	**0.0**
16	5397	60	[illegible]8.8	20.9	37.5	15.8	37.5	20.0	37.4	17.0	**0.0**	**0.0**
17	7397	20	0.4	5.5	6.1	6.0	4.9	6.7	4.36	7.4	**0.0**	**0.0**
18	7397	30	7.[illegible]	9.7	8.1	7.3	6.6	7.3	6.1	6.9	**0.0**	**0.0**
19	7397	40	4[illegible]	39.6	45.6	40.1	45.3	39.8	43.7	39.0	**0.0**	**0.0**
20	7397	50	34.1	26.2	32.7	26.6	33.2	26.9	33.2	25.7	**0.0**	**0.0**
21	7397	60	36.[illegible]	27.7	36.8	27.4	37.0	27.8	37.5	27.7	**0.0**	**0.0**
	Average		3[illegible]	30.7	22.8	23.9	31.8	30.0	25.3	22.4	**1.7**	**0.07**

From table 6, we can see that the percentage deviation of NGA is smaller than the ones of other four algorithms, and it shows that the proposed algorithm NGA can show superiority over the SAGA, HCGA, GAG and GA in term of solution quality.

The table 7 shows that NGA can show better solution quality than HHGA, GASA, GAHC and GAQSA for large scale CBTSP.

The following tables are the experiments results of the algorithms for large scale CBTSP by running 180s as the stopping condition.

In table 8 and table 9, GA, GAG and HCGA are all from the literatures [1-2]; the improved genetic algorithm SAGA is from the literature [1]; the modified genetic algorithms HHGA, GASA and GAHC respectively refer to the literature [22], literature [23] and literature [24]; the improved genetic algorithm GAQSA refers to the literature [25]; NGA is the proposed algorithm.

Table 8 the solution quality of the five algorithms for large scale CBTSP with running 180s as the stopping condition (Unit: km)

Instance	***n***	***m***	**GA**		**GAG**		**HCGA**		**SAGA**		**NGA**	
Large			Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean
1	2461	3	3782.0	3903.3	1152.0	1642.3	1345.0	1565.7	1132.0	[illegible]	**803.0**	**829.2**
2	2461	6	3869.0	3914.9	**1378.0**	2199.1	1760.0	2242.6	1356.0	1570.3	1395.0	**1490.1**
3	2461	12	3831.0	**3925.8**	2428.0	2965.3	2316.0	2993.7	[illegible]	[illegible]	1813.0	**2082.7**
4	2461	24	3901.0	3961.5	2814.0	3357.5	2771.0	3201.9	[illegible]	[illegible]	2442.0	**2577.0**
5	2461	30	3906.0	3951.6	3064.0	3280.5	2639.0	3187.9	[illegible]	2932.3	2555.0	**2701.1**
6	3461	3	4109.0	4200.4	1497.0	1897.3	1396.0	1932.3	1428.0	1828.0	**1209.0**	**1268.1**
7	3461	6	4155.0	4243.4	1870.0	2239.9	2032.0	2295.7	1796.0	2143.7	**1610.0**	**1853.3**
8	3461	12	4146.0	4196.4	2663.0	3135.0	2626.0	2988.3	**2005.0**	2536.5	2171.0	**2360.2**
9	3461	24	4124.0	4195.6	3603.0	3664.4	3335.0	3665.4	**2493.0**	3001.5	2764.0	**2943.2**
10	3461	30	4155.0	4193.9	3333.0	3663.2	3288.0	[illegible]	**2626.0**	3160.8	2859.0	**3123.1**
11	3461	40	4186.0	4245.2	3325.0	3725.9	3609.0	[illegible]	**3139.0**	3478.3	3132.0	**3271.0**
12	5397	20	728821.0	736354.9	703417.0	723289.7	677681.0	714741.2	651856.0	674221.4	**621091.0**	**649456.4**
13	5397	30	740720.0	751998.6	719075.0	743732.1	732496.0	[illegible]	675320.0	703960.5	**671429.0**	**692751.0**
14	5397	40	426569.0	434040.2	377954.0	384240.3	377937.0	[illegible]	377201.0	381100.4	**194797.0**	**259629.1**
15	5397	50	428986.0	434471.7	379344.0	393096.9	378736.0	[illegible]	377168.0	378335.9	**273101.0**	**317077.4**
16	5397	60	430699.0	435056.2	379385.0	400887.2	[illegible]	[illegible]	377149.0	379503.2	**283679.0**	**324001.2**
17	7397	20	689050.0	699144.1	603540.0	621953.6	587012.0	609802.3	**528691.0**	601227.8	559358.0	**567024.4**
18	7397	30	731807.0	736214.7	614787.0	662048.2	[illegible]	653935.7	602838.0	636154.6	**572646.0**	**595714.3**
19	7397	40	459451.0	463096.1	386784.0	390598.2	[illegible]	390747.5	379482.0	386187.3	**265908.0**	**277218.5**
20	7397	50	462300.0	463566.5	384435.0	390281.1	384424.0	388876.4	381267.0	387428.2	**281086.0**	**298361.5**
21	7397	60	461342.0	463348.3	386784.0	390239.6	[illegible]	392003.0	378147.0	385675.3	**281082.0**	**311882.9**

Table 9 the solution quality of the five algorithms for large scale CBTSP with running 180s as the stopping condition (Unit: km)

Instance	***n***	***m***	**HHGA**		**GASA**		**GAHC**		**GAQSA**		**NGA**	
Large			Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean
1	2461	3	1393.0	1717.8	970.0	[illegible]	1196.0	1826.3	1384.0	1664.3	**803.0**	**829.2**
2	2461	6	1991.0	2274.0	[illegible]	[illegible]	1735.0	2244.2	1612.0	1989.2	**1395.0**	**1490.1**
3	2461	12	2386.0	2896.6	**1424.0**	[illegible]	2414.0	2890.8	2235.0	2341.9	1813.0	**2082.7**
4	2461	24	2672.0	3173.5	**1883.0**	2802.2	2745.0	3230.6	2308.0	2589.3	2442.0	**2577.0**
5	2461	30	2820.0	3199.1	**[illegible]**	2812.4	2874.0	3217.8	2324.0	**2527.6**	2555.0	2701.1
6	3461	3	1712.0	2084.6	[illegible]	1764.8	1512.0	1956.7	1508.0	1830.1	**1209.0**	**1268.1**
7	3461	6	1830.0	2328.0	1799.0	2096.1	1938.0	2237.3	2027.0	2202.4	**1610.0**	**1853.3**
8	3461	12	2588.0	[illegible]	**2159.0**	2614.7	2882.0	3086.0	2264.0	2515.9	2171.0	**2360.2**
9	3461	24	3354.0	3566.2	**[illegible]**	3020.6	3324.0	3641.5	2517.0	3015.5	2764.0	**2943.2**
10	3461	30	3011.0	[illegible]	3098.0	3305.6	3111.0	3514.7	**2754.0**	3172.1	2859.0	**3123.1**
11	3461	40	3373.0	[illegible]	**2799.0**	3427.4	3346.0	3697.4	3078.0	3374.7	3132.0	**3271.0**
12	5397	20	695860.0	[illegible]	665442.0	689173.5	700339.0	717936.7	675930.0	688256.5	**621091.0**	**649456.4**
13	5397	30	716078	741008.1	679130.0	706743.8	709360.0	740032.9	674711.0	707437.5	**671429.0**	**692751.0**
14	5397	40	[illegible]	[illegible]	377138.0	379476.2	379344.0	382518.7	377142.0	378537.5	**194797.0**	**259629.1**
15	5397	50	378800.0	[illegible]	375260.0	379197.0	377840.0	384397.6	375496.0	379313.3	**273101.0**	**317077.4**
16	5397	60	[illegible]	[illegible]	377844.0	383271.6	379410.0	389680.2	376825.0	381004.5	**283679.0**	**324001.2**
17	7397	20	[illegible]	621945.7	588677.0	614833.5	599363.0	629677.5	607557.0	632747.0	**559358.0**	**567024.4**
18	7397	30	[illegible]	656876.8	602168.0	642598.7	603003.0	647381.0	608134.0	641923.1	**572646.0**	**595714.3**
19	7397	40	[illegible]	386670.4	379482.0	385773.6	383732.0	388376.7	372940.0	383990.7	**265908.0**	**277218.5**
20	7397	50	[illegible]	387736.2	381264.0	384867.0	385547.0	389602.7	375989.0	383986.7	**281086.0**	**298361.5**
21	7397	60	[illegible]	388113.9	379468.0	387324.5	383588.0	387633.4	384107.0	387665.3	**281082.0**	**311882.9**

In table 8 and table 9, they are the experiment results of the algorithms for CBTSP by running 180s as the stopping condition. The mean is also the average solution quality of the algorithms running 10 times for the problem. From the data, it shows that NGA can show better solution quality than the compared modified genetic algorithms.

The following figure 11 and figure 12 are the mean solution quality of the five algorithms for solving CBTSP by running 180s as the stopping condition.

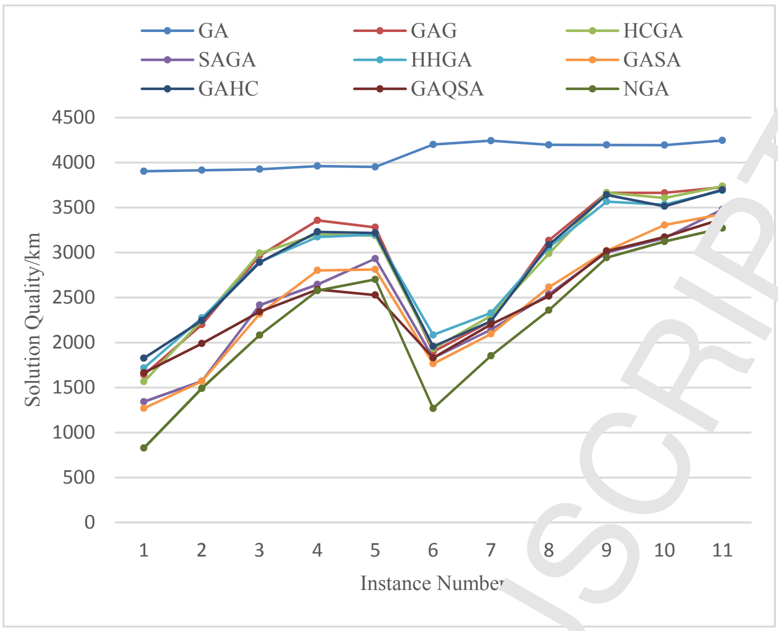

Fig.11. the average solution quality of the algorithms for large scale CBTSP (Unit: km)

In the figure 11, the lateral axis represents the order number of the instance. For example, the number order 6 corresponds to *n*=3461 and *m*=3, the number order 21 is the corresponding data with *n*=7397 and *m*=60; vertical axis is the mean solution quality of the algorithms for the problem.

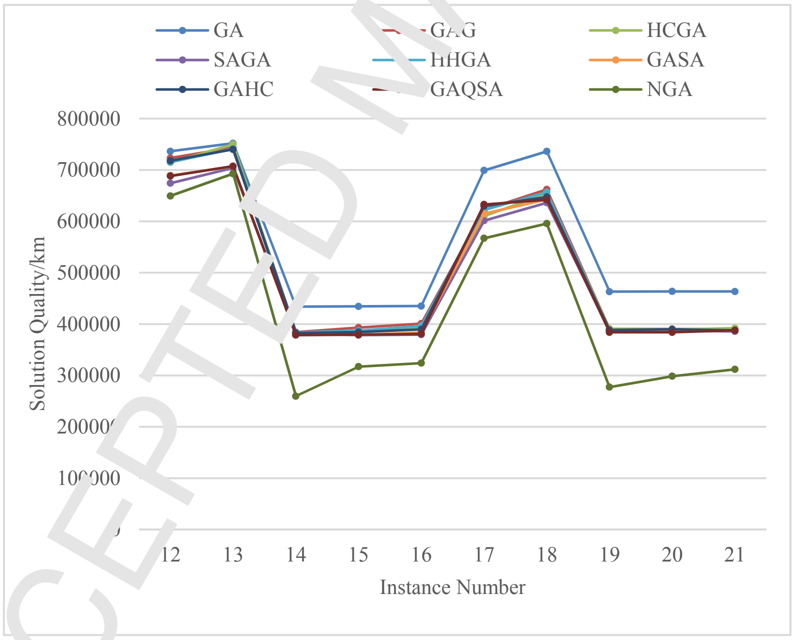

Fig.12. the average solution quality of the algorithms for large scale CBTSP (Unit: km)

The lateral axis of figure 12 means the order number of instance data, and the vertical axis is the mean solution quality of the algorithms for CBTSP.

The figure 11 and figure 12 show that NGA can demonstrate better mean solution quality than the modified genetic algorithms for solving large scale CBTSP.

The below tables are the percentage deviation of the algorithms for CBTSP by running 180s as stopping condition.

Table 10 the percentage deviation of the five algorithms for large scale CBTSP with running 180s as the stopping condition

Instance	***n***	***m***	**GA**		**GAG**		**HCGA**		**SAGA**		**NGA**	
Large			PD_{best}	PD_{av}	PD_{best}	PD_{av}	PD_{best}	PD_{av}	PD_{best}	PD_{av}	PD_{best}	PD_{av}
1	2461	3	370.9	370.7	43.4	98.0	67.4	88.8	40.9	61.9	**0.0**	**0.0**
2	2461	6	185.3	162.7	1.6	47.5	29.7	50.4	0.0	5.3	[illegible]	**0.0**
3	2461	12	122.6	88.4	41.0	42.3	34.5	43.7	0.0	[illegible]	5.3	**0.0**
4	2461	24	123.0	53.7	60.8	30.2	58.4	24.2	0.0	2.6	39.6	**0.0**
5	2461	30	64.6	46.2	29.1	21.4	11.2	18.0	0.0	[illegible]	7.7	**0.0**
6	3461	3	239.8	231.2	23.8	49.6	15.4	52.3	18.1	44.1	**0.0**	**0.0**
7	3461	6	243.6	128.9	54.6	20.8	68.0	23.8	48.5	[illegible]	33.1	**0.0**
8	3461	12	106.7	77.7	32.8	32.8	30.9	26.6	0.0	7.46	8.2	**0.0**
9	3461	24	65.4	42.5	44.5	24.5	33.7	24.5	0.0	1.9	10.8	**0.0**
10	3461	30	58.2	34.2	26.9	17.2	25.2	15.4	0.0	1.2	8.8	**0.0**
11	3461	40	47.9	29.7	17.5	13.9	27.5	14.2	10.9	6.3	10.7	**0.0**
12	5397	20	17.3	13.3	13.25	11.3	9.1	10.0	4.9	3.8	**0.0**	**0.0**
13	5397	30	10.3	8.5	7.0	7.3	9.0	8.2	0.5	1.6	**0.0**	**0.0**
14	5397	40	118.9	67.1	94.0	47.9	94.0	46.2	[illegible]	46.7	**0.0**	**0.0**
15	5397	50	57.0	37.0	38.9	23.9	38.6	[illegible]	[illegible]	19.3	**0.0**	**0.0**
16	5397	60	51.8	34.2	33.7	23.7	33.3	[illegible]	32.9	17.1	**0.0**	**0.0**
17	7397	20	30.3	23.3	14.1	9.6	11.0	7.5	0.0	6.0	5.8	**0.0**
18	7397	30	27.7	23.5	7.3	11.1	10.2	[illegible]	5.2	6.7	**0.0**	**0.0**
19	7397	40	72.7	67.0	45.4	40.8	44.3	40.9	42.7	39.3	**0.0**	**0.0**
20	7397	50	64.4	55.3	36.7	30.8	36.7	30.3	35.6	29.8	**0.0**	**0.0**
21	7397	60	64.1	48.5	37.6	25.1	[illegible]	25.6	34.5	23.6	**0.0**	**0.0**
	Average		102.0	78.3	33.5	30.0	34.5	28.7	19.3	17.4	**6.3**	**0.0**

In table 10, it shows that the best solution percentage deviation PD_{best} and average best solution percentage deviation PD_{av} of NGA is the smallest in the five algorithms, which means that NGA can demonstrate superiority over SAGA, HCGA, GAG and GA in term of solution quality.

Table 11 the percentage deviation of the five algorithms for large scale CBTSP with running 180s as the stopping condition

Instance	***n***	***m***	**HHGA**		**GASA**		**GAHC**		**GAQSA**		**NGA**	
Large			PD_{best}	PD_{av}	PD_{best}	PD_{av}	PD_{best}	PD_{av}	PD_{best}	PD_{av}	PD_{best}	PD_{av}
1	2461	3	73.4	107.1	[illegible]	53.0	48.9	120.2	72.3	100.7	**0.0**	**0.0**
2	2461	6	42.7	52.6	**0.0**	5.2	24.3	50.6	15.5	33.4	**0.0**	**0.0**
3	2461	12	67.5	39.0	**0.0**	11.2	69.5	38.8	56.9	12.4	27.3	**0.0**
4	2461	24	41.9	[illegible]	**0.0**	8.7	45.7	25.3	22.5	0.4	29.6	**0.0**
5	2461	30	47.2	26.5	**0.0**	11.2	50.0	27.3	21.3	**0.0**	33.4	6.8
6	3461	3	41.6	[illegible]	[illegible]	39.1	25.0	54.3	24.7	44.3	**0.0**	**0.0**
7	3461	6	13.6	25.6	[illegible]	13.1	20.3	20.7	25.9	18.8	**0.0**	**0.0**
8	3461	12	[illegible]	29.5	**0.0**	10.7	33.4	30.7	4.8	6.5	0.5	**0.0**
9	3461	24	36.0	[illegible]	**0.0**	2.6	34.7	23.7	2.0	2.4	12.0	**0.0**
10	3461	30	[illegible]	13.1	12.4	5.8	12.9	12.5	**0.0**	1.5	3.8	**0.0**
11	3461	40	20.5	12.7	**0.0**	4.7	19.5	13.0	9.9	3.1	11.8	**0.0**
12	5397	20	[illegible]	10.1	7.1	6.1	12.7	10.5	8.8	5.9	**0.0**	**0.0**
13	5397	30	6.6	6.9	1.1	2.0	5.6	6.8	0.4	2.1	**0.0**	**0.0**
14	5397	40	94.5	47.8	93.6	46.1	94.7	47.3	93.6	45.7	**0.0**	**0.0**
15	5397	50	[illegible]	22.0	37.4	19.5	38.3	21.2	37.4	19.6	**0.0**	**0.0**
16	5397	60	[illegible]	22.3	33.1	18.2	33.7	20.2	32.8	17.5	**0.0**	**0.0**
17	7397	20	5.1	9.6	5.2	8.43	7.1	11.0	8.6	11.5	**0.0**	**0.0**
18	7397	30	9.6	10.2	5.1	7.8	5.3	8.6	6.1	7.7	**0.0**	**0.0**
19	7397	40	44.2	39.4	42.7	39.1	44.3	40.0	40.2	38.5	**0.0**	**0.0**
20	7397	50	36.5	29.9	35.6	28.9	37.1	30.5	33.7	28.6	**0.0**	**0.0**
21	7397	60	36.6	24.4	35.0	24.1	36.4	24.2	36.6	24.2	**0.0**	**0.0**
	Average		34.8	30.3	17.2	17.4	33.3	30.3	26.4	20.2	**5.6**	**0.3**

In table 11, it shows that NGA can demonstrate better solution quality than the compared modified genetic algorithms HHGA, GASA, GAHC and GAQSA.

4.3 Discussion

In intelligent transport systems and multiple tasks cooperation, many real-world problems can be modeled by CBTSP, the scale of constructed model is easy to tend to large scale, thus it is significant to study large scale CBTSP, however, the traditional modified genetic algorithms, such as GAG, HCGA and SAGA, are easy to fall into local optimum, in order to improve it, the NGA is proposed for this problem. For the small scale and medium scale CBTSP, the experiments show that NGA has no obvious superiority over the compared genetic algorithms, while the city number is more than 2000, the traditional modified genetic algorithms are easy to fall into local optimum, NGA can display strong global search ability, it can demonstrate obvious superiority over the compared algorithms.

The NGA and the modified genetic algorithms for large scale CBTSP are shown in table 4 and table 5 by running 60s as the stopping condition, the figure 9 and figure 10 are the average solution quality of the nine algorithms for this problem, which is made based on the average solution of the former two tables, the percentage deviation of table 6 and table 7 is made based on the table 4 and table 5. The tables 4-7 and figures 9-10 show that NGA has better solution quality than the compared modified genetic algorithms. Table 8 and table 9 are experiments results of the algorithms for large scale CBTSP by running 180s as the stopping condition, the following figures 11-12 and tables 10-11 are made based on table 8 and table 9. Tables 8-11 and figures 11-12 display that NGA can demonstrate better solution than the compared genetic algorithms.

In the mentioned tables and figures, it shows that NGA can demonstrate better solution quality than the compared modified genetic algorithms GAG, HCGA, SAGA, HFGA, GASA, GAHC and GAQSA for large scale CBTSP. HCGA and SAGA have the similar solution quality, and GA displays the worst solution quality in the several algorithms. In term of the solution quality, the improved GAs such GAG, HCGA and SAGA have better performance than the basic GA. NGA has the best solution quality in the nine algorithms for solving the large-scale problem.

From the above experiments, it shows NGA can not only show better best solution quality than the former compared algorithms in most of instances for large scale CBTSP, but also display obvious superiority over the compared modified genetic algorithms in term of average solution quality. NGA is a new swarm intelligence algorithm which can be used in optimization problems such as planning and combination optimization problem. By the CBTSP experiments, it shows that the NGA is effective for large scale CBTSP, and can demonstrate superiority over the compared genetic algorithms.

5 Conclusion and future works

CBTSP is one of the combination optimization problems, which is from the applications where multiple salesmen work cooperatively in the workspaces that partially overlap with each other. The paper provides a new model called CBTSP, which can model real-world problems, and proposes a novel genetic algorithm for solving the problem. The extensive experiments show that NGA can demonstrate better performance than the compared modified genetic algorithms for large scale CBTSP in term of solution quality.

The limitations of our works are as follows: although the city number of CBTSP is more than 7000, the scale is still limited; the given applications of CBTSP is not enough in this paper. The next possible works could be focused on the points: on the one hand, studying more advanced algorithms for larger scale CBTSP is a possible research area, the expected algorithms should show good performance in term of solution quality or solving speed; on the other hand, exploring and studying more applications of CBTSP model is another possible research work, and we can also use the proposed algorithm to solve other combinational optimization problems. In addition, the new learning strategies, such as multi-tasking learning [27-29], reinforcement learning [30-33], social learning [34-36], or self-learning [37-38], should be introduced in the used algorithms for further improving their performance.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant NO.61672024, No. 61170305; Shandong Provincial Natural Science Foundation of China under Grant NO. ZR201808190052.

Reference

[1] J. Li, M. Zhou, Q. Sun, X. Dai, and X. Yu, Colored traveling salesman problem, IEEE Transactions on Cybernetics, 2015, 45(11):2390-2401.

[2] J. Li, S. Qiru, M. C. Zhou, et al, Colored traveling salesman problem and solution, the 19th World Congress International Federation of Automatic Control, Cape Town, South Africa, 2014, pp. 24-29.

[3] X.S. Dong, W.Y. Dong and Y.F. Wang, Hybrid algorithms for multi-objective balanced traveling salesman problem, Journal of Computer Research and Development, 2017, 54(8):1751-1762.

[4] E. Ardjmand, W.A. Yung, G.R. Weckman, et al, Applying genetic algorithm to a new bi-objective stochastic model for transportation, location, and allocation of hazardous materials, Expert Systems With Applications, 2016, 51: 49-58.

[5] N. Metawa, M.K. Hassan, M. Elhoseny, Genetic algorithm based model for optimizing bank lending decisions, Expert Systems With Applications, 2017, 80:75-82.

[6] P. Ghosh, M. Mitchell, J.A. Tanyi, A. Y. Hung, Incorporating priors for medical image segmentation using a genetic algorithm, Neurocomputing, 2016, 195:181-194.

[7] X.S. Dong, W.Y. Dong and Y.L. Cai, Hybrid algorithm for colored bottleneck traveling salesman problem, Journal of Computer Research and Development, 2018, 55(11):2372-2385.

[8] Y. Ding, X. Fu, Kernel-based fuzzy c-means clustering algorithm based on genetic algorithm, Neurocomputing, 2016, 188:233-238.

[9] Y. Ar, E. Bostanci, A genetic algorithm solution to the collaborative filtering problem, Expert Systems With Applications, 2016, 61:122-128.

[10] D. Martin, J. A. Fdez, A. Rosete, F. Herrera, NICGAR: A niching genetic algorithm to mine a diverse set of interesting quantitative association rules, Information Science, 2016, 355-356:208-228.

[11] H. Höglund, Tax payment default predication using genetic algorithm-based variable selection, Expert Systems With Applications, 2017, 88:368-375.

[12] Y. Zelenkov, E. Fedorova, D. Chekrizov, Two-step classification method based on genetic algorithm for bankruptcy forecasting, Expert Systems With Applications, 2017, 88:393-401.

[13] H. HAKLI, H. UĞUZ, A novel approach for automated land partitioning using genetic algorithm, Expert Systems With Applications, 2017, 82:10-18.

[14] X.Y. Zhang, J. Zhang, Y.J. Gong, et al, Kuhn-munkres parallel genetic algorithm for the set cover problem and its application to large-scale wireless sensor networks, IEEE Transactions on Evolutionary Computation, 2016, 20(5):695-710.

[15] T. Friedrich, T. Kötzing, M.S. Krejca, A.M. Sutton, The compact genetic algorithms is efficient under extreme Gaussian noise, IEEE Transactions on Evolutionary Computation, 2017, 21(3):477-490.

[16] M.A. Rashid, F. Khatib, M.T. Hoque, A. Sattar, An enhanced genetic algorithm for ab initio protein structure prediction, IEEE Transactions on Evolutionary Computation, 2016, 20(4):627-644.

[17] T.M. Lakshmi, A. Martin, V.P. Venkatesan, A genetic bankrupt ratio analysis tool using a genetic algorithm to indentify influencing financial ratios, IEEE Transactions on Evolutionary Computation, 2016, 20(1):38-51.

[18] W.Y. Dong, W.S. Zhang, R.G Yu, Convergence and runtime analysis of ITÖ algorithm for one class of combinatorial optimization, Chinese Journal of Computers, 2011, 34(4): 636-646.

[19] Y.F. Wang, W.Y. Dong, X.S. Dong, A novel ITÖ Algorithm for influence maximization in the large-scale social networks, Future Generation Computer Systems, 2018, 88: 755-763.

[20] A. P. Engelbrecht, Fitness function evaluations: a fair stopping condition? Proceedings of Swarm Intelligence Symposium on Swarm Intelligence, Orlando, FL, USA: IEEE, 2014, pp.1-8.

[21] M. Mernik, S. H. Liu, D. Karaboga, et al, On clarifying misconceptions when comparing variants of the artificial bee colony algorithm by offering a new implementation, Information Science, 2015, 291:115-127.

[22] S. C. Su, C. J. Lin, C. K. Ting, An efficient hybrid of hill-climbing and genetic algorithm for 2D triangular protein structure prediction, 2010 IEEE International Conference on Bioinformatics and Biomedicine Workshops, Hong Kong, 2015, 51-56.

[23] H. H. Örkcü, Subset selection in multiple linear regression models: A hybrid of genetic and simulated annealing algorithms, Applied Mathematics and Computation, 2013, 219:11018-11028.

[24] S. Jun, J. Park, A hybrid genetic algorithm for the hybrid flow shop scheduling problem with nighttime work and simultaneous work constraints: A case study from the transformer industry, Expert Systems with Applications, 2015, 42: 6196-6204.

[25] R. Sonmez, Ö. H. Bettemir, A hybrid genetic algorithm for the discrete time–cost trade-off problem, Expert Systems with Applications, 2012, 39:11428-11434.

[26] H.G. Zhang, J. Zhou, Dynamic multicale region search algorithm using vitality selection for traveling salesman problem, Expert Systems With Applications, 2016, 60:81-95.

[27] L. Zhao, Q. Sun, J. P. Ye, Multi-task learning for spatio-temporal event forecasting, ACM SIGKDD Conference on Knowledge Discovery & Data Mining (SIGKDD2015), 2015, pp.1503-1512.

[28] C. S. Li, J.C. Yan, F. Wei, et al, Self-paced multi-task learning, Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017), 2017, pp. 2175-2181.

[29] A.A. Liu, Y.T. Su, W.Z. Nie, M. Kankanhalli, Hierarchical clustering multi-task learning for joint human action grouping and recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 39(1):102-113.

[30] V.Mnih, K.Kavukcuoglu, D. Silver, et al, Human-level control through deep reinforcement learning, Nature, 2015, 518(7540):529-33.

[31] F. C.Ghesu, B. Georgescu, Y.F. Zheng, et al, Multi-scale deep reinforcement learning for real-time 3D-landmark detection in CT Scans, IEEE Transactions on Pattern Analysis and Machine Intelligence, DOI: 10.1109/TPAMI.2017.2782687.

[32] F.Belletti, D.Haziza, G. Gomes, et al, Expert level control of ramp metering based on multi-task deep reinforcement learning, IEEE Transactions on Intelligent Transportation Systems, 2018,19(4):1198-1207.

[33] D. Mar, G.G. Aleny, C. Torras, Relational reinforcement learning with guided demonstrations, Artificial Intelligence, 2017,247, 295-312.

[34] E.Karataev, V.Zadorozhny, Adaptive social learning based on crowdsourcing, IEEE Transactions on Learning Technologies, 2017, 10(2): 128-139.

[35] L.Molleman, P. Berg, F.J. Weissing, Consistent individual differences in human social learning strategies, Nature Communications, 5:3570 doi: 10.1038/ncomms4570 (2014).

[36] R. Cheng, Y.C.Jin, A social learning particle swarm optimization algorithm for scalable

optimization, Information Sciences, 2015, 291:43-60.

[37] B. Lu, M. Coombes, B.B. Li,et al, Improved situation awareness for autonomous taxiing through self-learning, IEEE Transactions on Intelligent Transportation Systems, 2016, 17(12): 3553-3564.

[38] J.H. Zhong, Y.S. Ong, W.T. Cai, Self-Learning gene expression programming, IEEE Transactions on Evolutionary Computation, 2016, 20(1): 65-80.

Xueshi Dong is currently assistant professor in College of Computer Science and Technology, Qingdao University. He ever pursued Ph.D degree in Computer School, Wuhan University from 2013 to 2018. He worked in Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences from 2011 to 2012, and he also visited Institute of Computing Technology, Chinese Academy of Sciences and Department of Computer Science and Technology, Peking University from 2015 to 2016. His current research interests include artificial intelligence, intelligent computing and simulation optimization.

Yongle Cai received M.S. degree in Computer School, Wuhan University. His main research interests include intelligent computing and simulation optimization.

Xueshi Dong

Yongle Cai

Highlights:

This paper provides a new colored balanced traveling salesman problem (CBTSP) model, which can be used to model the optimization problems with the partially overlapped workspace.

The paper extends the scale of the model to large scale in which the city number is more than 1000, and studies the large scale optimization for CBTSP.

A novel genetic algorithm is proposed for large scale CBTSP, and the experiments show the superiority of the proposed algorithm.