Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Colored Traveling Salesman Problem and Solution

Jun Li[1, 2], Qirui Sun[1, 2], MengChu Zhou[3], Xiaolong Yu[1, 2], and Xianzhong Dai[1, 2]

1. Key Laboratory of Measurement and Control of CSE, Ministry of Education, Nanjing 210096, China
2. School of Automation, Southeast University, Nanjing 210096 China (e-mail: {j.li, xzdai}@seu.edu.cn; {sunqr, xly19900107}@163.com)
3. New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu)

Abstract: Multiple Traveling Salesman Problem (MTSP) is an important combinatorial optimization problem. However, it is applicable to only the cases in which multiple executing individuals (traveling salesman) share the common workspace (city set). It cannot be used to handle many multi-machine engineering systems where multiple machines' workspaces are not the same and partially overlap with each other. This paper proposes and formulates a new MTSP called colored traveling salesman problem (CTSP). Each of its salesmen is assigned a private city set and all salesmen share a public city set. Every set of cities is colored differently. To solve CTSP, we present two improved genetic algorithms (GA) by combining the classic one with a greedy algorithm and hill-climbing one to achieve better performance. Finally, the algorithms are applied and compared through a case study. The result shows that the hill-climbing GA enjoys the best performance among the investigated ones.

Keywords: TSP, MTSP, Modelling, Genetic Algorithm, Greedy Algorithm, Hill-climbing Algorithm

1. INTRODUCTION

A multiple traveling salesman problem (MTSP) generalized from a traveling salesman problem (TSP) is a well-known combinatorial optimization problem. It aims to determine a family of tours with minimal total cost for multiple salesmen to visit each city exactly once within a given set and eventually return to the home city. MTSP and TSP arise in a variety of applications that require addressing scheduling, planning, routing, and/or sequencing issues. Application examples of TSP in machine scheduling and sequencing, and vehicle routing can be founded in (Gutin and Punnen, 2002). Other work reports additional applications in circuit wiring (Hirogaki, *et al.,* 2005) and in statistical data analysis including ordering and clustering objects, e.g., gene ordering in (Ray, *et al.,* 2007) and protein clustering in (Johnson and Liu, 2007). The most comprehensive survey on the applications of MTSP is given in (Bektas, 2006). Carter and Ragsdale (2006) stress its use in pre-print insert advertisement scheduling. A similar application in hot rolling scheduling is reported in (Tang, *et al.,* 2000). Autonomous robot or vehicle motion planning (Basu, *et al.,* 2000, and Ryan, *et al.,* 1998) represents other types of its applications. Saleh and Chelouah (2004) apply it to satellite surveying system design. Toth and Vigo (2002) investigate a vehicle routing problem as the generalization of TSP. Cheong and White (2012) have investigated how to dynamically determine a tour for TSP based on real-time traffic congestion data.

In essence, MTSP is an abstraction of the practical problems in which multiple executing individuals (traveling salesmen) are involved and they share the common workspace (city set). In other words, all the cities of MTSP are identical for each salesman, i.e., each city can be visited by any salesman. However, not all executing individuals have the same workspace in some application problems. Take the scheduling of a multi-machine engineering system (MES) as an example. The workspaces of individual machines are not the same but overlap partially with each other. Thus, each machine has to perform not only the operations independently in its private workspace, but also complete all the operations with other machine(s) together in the overlapped workspace. A typical MES, i.e., a dual-bridge waterjet cutting machine tool, is illustrated in Fig. 1.

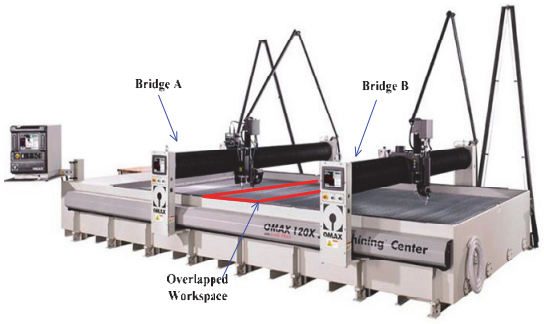

Fig. 1. Dual-bridge Waterjet Cutting Machine Tool

It consists of two independent bridge systems. Their cutting areas have an overlapped section, i.e., the marked area on the workbench with a red box, so as to prevent the presence of cutting dead zone. Thus, the overlapped area allows both bridges to enter and the two areas out of it are their exclusive cutting areas.

Due to the partially overlapped workspaces, a scheduling method for MTSP cannot be simply used to schedule MES.

Copyright © 2014 IFAC

On the other hand, the basic elements of such a problem, i.e., individuals, operations, and workspaces, are still similar to the salesmen, city visits and city set of TSP, respectively. The difference lies in that each individual (salesman) of the former not only has a private workspace (city set) but also shares a common workspace with others. To distinguish the different cities, we define a new multiple traveling salesman problem by coloring the cities, called Colored TSP (CTSP). CTSP frequently arises in real-life applications where some closely dependent relations between the salesmen and the cities must be obeyed when one determines a solution. It is a significant problem in theory and practice. We call the same problem as MTSP* by Li, *et al.* (2013) and present a genetic algorithm (GA) solution. However, CTSP has not been formulated in a mathematically rigorous way. This paper formally defines CTSP and improves the prior method in (Li, *et al.,* 2013) by combining GA with Greedy Algorithm and Hill-climbing Algorithm.

Next, CTSP is formulated in Section 2. Section 3 presents two improved GAs. Section 4 gives a case study with the comparison results. The paper is concluded in Section 5.

2. DEFINITION AND FORMULATION OF CTSP

2.1 CTSP Definition

Let $n, m \in \mathbf{Z} = \{1,2,3,...\}$ and $m < n$. CTSP aims to determine a family of tours with the minimal total cost for *m* salesmen to visit *n* cities exactly once given public and private ones, and eventually return to the home city (depot). Let V_i, $i \in \mathbf{Z}_m = \{1,2,...,m\}$ be the private city set assigned to the *i*-th salesman and U_j, $j \in \mathbf{Z}_r, r \geq 1$ be the *j*-th shared city set, and W_i be the accessible city set of the *i*-th salesman, i.e., the union of sets of *i*-th salesman's private cities and shared cities.

The city sets meet the following constraints, given $i \neq j$:

$$U_i \cap U_j = \varnothing \ , \forall i, j \in \mathbf{Z}_r \tag{1}$$

$$W_i \not\subset W_j \ \text{ and } W_j \not\subset W_i, \forall i, j \in \mathbf{Z}_m \tag{2}$$

There are various cases of the intersections among the accessible city sets in CTSP. A common one is that there is only one common city set shared by *m* salesmen, as shown in Fig. 2.

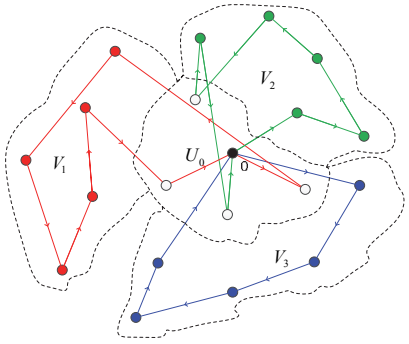

Fig. 2. Example of CTSP

The nodes in the areas V_1, V_2, and V_3 represent the private cities of salesmen 1, 2, and 3. They have the only shared city set V_0.

2.2 0-1 Integer Programming Model

CTSP is formulated over a complete digraph $G = (V, E)$, where the vertex set $V = \{0,1,2,...,n-1\}$ corresponds to the cities and each edge in $(i, j) \in E$, $i \neq j$, is associated with a weight ω_{ij} representing a visit cost (distance) between two cities *i* and *j*. The vertex 0 represents the home city (depot). Let $\tilde{V} = V \setminus \{0\}$. V is divided into *m*+1 sets, i.e., V_0, the public one, and V_i, the private ones of the salesmen for all $i \in \mathbf{Z}_m$. The objective of CTSP is to determine *m* Hamiltonian cycles or circuits on *G* with the least total cost such that any vertex of each private set is visited exactly once by the specified salesman and any vertex of the public set is visited by any salesman exactly once and eventually return to city 0. Of course, it allows each salesman to have an exclusive home city in their private set, like the waterjet cutting example shown in Fig. 1. However, this work will not discuss such cases.

Binary variable x_{ijk} =1, $i \neq j$, $i, j \in V$, and $k \in \mathbf{Z}_m$, if the *k*-th salesman passes through edge (i, j); and otherwise, $x_{ijk} = 0$. u_{ik} is the number of nodes visited on the *k*-th salesman's tour from the depot up to node *i*.

The integer programming model of CTSP is presented as follows.

$$\text{Minimize} \sum_{k=1}^{m} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \omega_{ij} x_{ijk} \tag{3}$$

$$\text{Subject to} \sum_{i=1}^{n-1} x_{0ik} = 1, \tag{4}$$

$$\sum_{i=1}^{n-1} x_{i0k} = 1, \ k \in \mathbf{Z}_m, \tag{5}$$

$$\sum_i \sum_j x_{ijk} = 0, \tag{6}$$

$$\sum_i \sum_j x_{jik} = 0, \ i \in V_k, j \in \tilde{V} \setminus (V_0 \cup V_k), k \in \mathbf{Z}_m, \tag{7}$$

$$\sum_i \sum_j x_{ijl} = 0, \tag{8}$$

$$\sum_i \sum_j x_{jil} = 0, \ \ i \in V_k, j \in V, i \neq j, k \neq l, l \in \mathbf{Z}_m, \tag{9}$$

$$\sum_{j=0}^{n-1} \sum_{k=1}^{m} x_{ijk} = 1, \tag{10}$$

$$\sum_{j=0}^{n-1}\sum_{k=1}^{m} x_{jik} = 1,\ i \in \tilde{V}, j \neq i, \tag{11}$$

$$\sum_{h} x_{jhk} = \sum_{i} x_{ijk},\ j \in V_0, i, h \in V_k \cup V_0, i \neq j \neq h, \tag{12}$$

$$u_{ik} - u_{jk} + n \times x_{ijk} \leq n-1,\ i, j \in \tilde{V}, j \neq i, k \in \mathbf{Z}_m, \tag{13}$$

Equations (4) and (5) require that every salesman start from and return to city 0, and (6) ensures that salesman k cannot start from his own exclusive city to visit a private city of other salesmen and (7) that another salesman is forbidden to visit a private city of salesman k from its own private city as well. Equations (8) and (9) ensure that salesman $l(\neq k)$ can neither start from a private city of salesman k nor return to it. Each city except city 0 can be visited exactly once as described by (10) and (11). Equation (12) represents that a public city can be visited by any salesman while (13) prohibits the formation of any sub-tour among nodes in $V \backslash \{0\}$.

Like MTSP, CTSP is also NP-hard. Moreover, the restriction on city colours makes its solution more difficult and time-consuming than that of MTSP. It is proven that the heuristics are faster and more efficient than the exact methods in the solution of MTSP with respect to the problem size. In many cases, however, the former cannot be guaranteed to obtain the optimal solution and are thus applicable to solve those cases in which good-quality solutions suffice. With this in mind, this paper presents GA for CTSP.

3. GENETIC ALGORITHMS FOR CTSP

3.1 GA and Its Limitation

Our prior work developed a basic GA to solve CTSP (Li, *et al.,* 2013). It represents a solution via dual chromosomes that are decimally coded, i.e., city and salesman chromosomes. Constraints (6)-(12) as a city assignment relation are taken account into the dual-chromosome coding where each city gene corresponds to a right salesman gene at the same position. Suppose that private cities of salesmen 1-3 are cities 1-2, 3-4, and 5-6, respectively, and the shared cities are cities 7-10.

Li, *et al.,* (2013) adopt the combination of Roulette Wheel method and Elitist strategy as the selection operation. Three pairs of compositions of the crossover and mutation operators are compared and the result shows that the performance of city crossover and city mutation (CCM) operator is the best. A city crossover operator is a modified partially matched crossover (PMX). Figure 3 shows a crossover process of a dual-chromosome with a single crossover of city chromosomes.

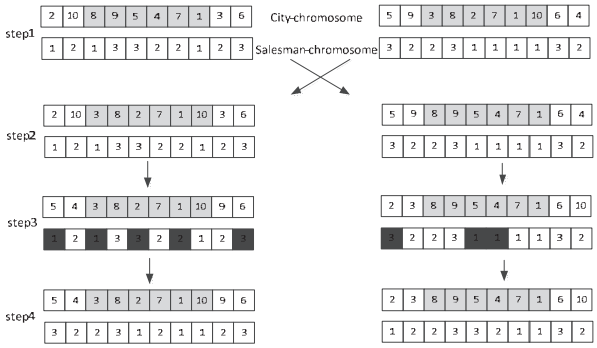

Fig. 3. Example of CC

In Step 1, given two parents, a section of a city individual is selected at random, and then its genes are swapped with those of another individual, thereby resulting in two new individuals as shown in Step 2. The mapping relationship of the selected sections in two city individuals is 8—3, 9—8, 5—2, 4—7, 7—1, and 1—10. Step 3 exchanges the redundant genes according to the selected section, and then finds that private cities 5, 3, 7, 1, and 6 in the left chromosome and cities 2, 5, and 4 in the right one are assigned to the wrong salesmen. Next, Step 4 reassigns the private cities to the correct salesmen and obtains two reasonable generations.

A city mutation (CM) process in a dual-chromosome is illustrated in Fig.4.

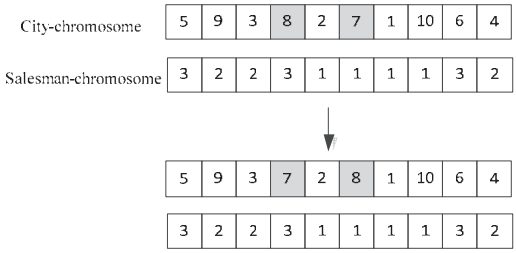

Fig. 4. Example of city mutation (CM)

First, the gene points of cities 8 and 7 are selected as swapping ones. After swapping, the city assignment relation is satisfied and the mutation is over.

The fitness function takes the value of the inverse of the total tour cost $f(x)$ equal to that of Eq. (3).

$$F(x) = \frac{1}{1+f(x)} \tag{14}$$

In (Li, *et al.,* 2013), the case study indicates that the evolution of GA is slow and it is easy to trap in a local optimum.

3.2 Greedy GA

The decision made by using Greedy Algorithm at each step may not reach the best in the global view but the local optimum. However, it can obtain the satisfactory solution rapidly because it avoids the great effort needed to exhaust all possibilities to find the optimal solution. We use it to

optimize the individuals of the initial population generated randomly at the first step of GA. Initial population of high quality will accelerate the population evolution of GA and reach satisfactory solution rapidly. We name this improved algorithm as a greedy GA.

With regard to CTSP, the criterion is defined as the shortest distance between two cities. Namely, a city will be selected as the next one that the corresponding salesman will visit once it is nearest to the current city in the visited sequence. It can optimize a solution by reordering its sequence. For example as shown in Fig. 5, the randomly generated visit sequence is 0→2→5→3→1→4→0. The sum of distances is 25+50+50+45+45+25=240. It can be optimized to be sequence $0 \to 3 \to 4 \to 5 \to 1 \to 2 \to 0$ by using the greedy algorithm. The improved distance is 22+25+27+35+25+25=159. Obviously, it is a better solution.

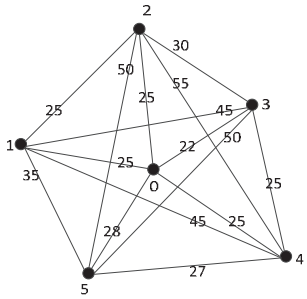

Fig. 5. Distances between cities

The generation process of the initial population in greedy GA is as follows.

Step 1: Determine if the number of individuals in the current initial population is equal to the set number N or not. If it is true, terminate the process; otherwise, go to the next step.

Step 2: Generate a city sequence randomly and assign the private cities to the specified salesman and the public cities to all the salesmen randomly. It results in individual *a.*

Step 3: Reorder the city sequence of a by the shortest distance criterion to minimize the visit cost and obtain individual a'.

Step 4: Detect if a' has already existed in the population or not. If so, go back to Step 2; otherwise, insert it into the population and go back to Step 1.

3.3 Hill-Climbing GA

Hill-climbing Algorithm utilizes neighbourhood search techniques to search, like hill-climbing, in a single direction that the quality of a solution is possible to be improved (Lim, *et al.*, 2006). Starting from an existing node, it generates a new solution with a method of neighbourhood point selection and compares it with the value of the existing node. If the former is larger, replaces the latter by the former; otherwise, return the latter and set it as the maximum. Repeat the process of climbing upward (to better solution) until the highest point is reached. The local search power of the algorithm is very strong and it is a common method used for the local optimum search.

GA in (Li, *et al.,* 2013) adopts the combinatorial selection strategy of elite reservation and roulette in (Sun, 2013). After a certain period of evolution, it may be trapped into a local optimum. To escape from it, the best individual of each generation can be optimized by using Hill-Climbing Algorithm. Specifically, if a better individual is obtained through hill-climbing, it replaces the original one; and otherwise, the original one remains in it. Note that the hill-climbing GA adopts Greedy Algorithm to optimize the initial population too.

The neighbourhood point selection impacts greatly on the hill-climbing algorithm and the paper adopts two point swapping. Given CTSP with $m(\geq 2)$ salesmen, it should select $2m$ genes by this selection strategy. The fitness must be recalculated after every time of gene swapping.

A hill-climbing GA includes the following steps:

Step 1: Determine if the i-th salesman performing the current swapping is the m-th salesman, i.e., $i=m$. If so, end this hill-climbing; otherwise, go to the next step.

Step 2: Select two city genes assigned to the i-th salesman, from the city chromosome of a. Swap them and obtain individual a' , and go to the next step.

Step 3: Determine if the value of fitness of a' is greater than that of a. If so, let $a = a'$; and otherwise, give up a' and keep a.

Step 4: Let $i = i + 1$, and return to Step 1.

The main procedure of hill-climbing GA is summarized in Fig. 6.

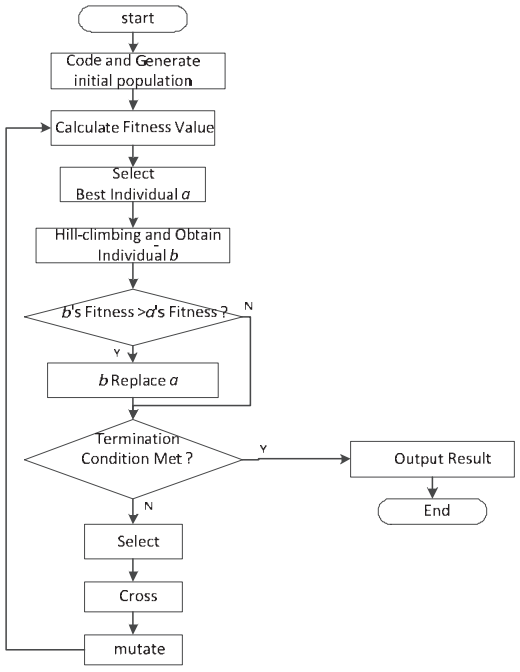

Fig. 6. Flowchart of Hill-climbing GA

4. CASE STUDY

A CTSP with $n = 51$ and $m = 4$ is shown in Fig. 7, where V_0 is the public city set (visit area) and V_1-V_4 are the private city sets (visit areas) of Salesmen 1-4, respectively. All the algorithms and processes are implemented in C++ on the platform Microsoft Visual Studio 2010. The computer used is Dell Inspiron620s having Windows 7 (32 bits) with CPU Intel Corei3 and 2GB RAM at 3.30GHz.

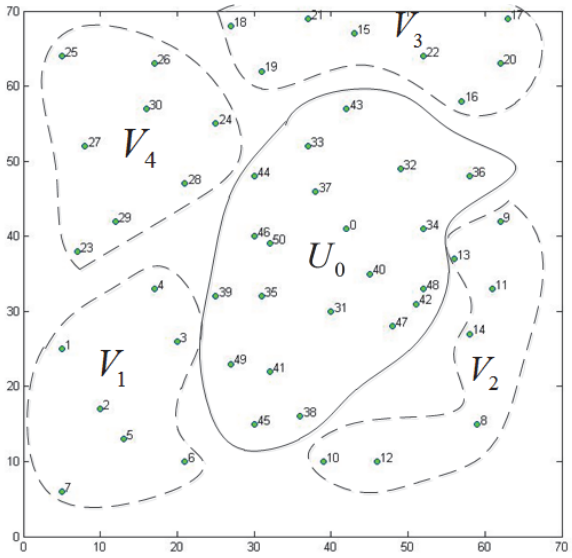

Fig. 7 CTSP and its city distribution

Next, the three algorithms GA, greedy GA, and Hill-climbing GA are applied to solve the problem and their performances are compared. We set the same parameters, i.e., the individual number of a population to be 30, crossover probability 0.7, and mutation probability 0.1. Each algorithm is run for five times and the maximum of epochs is 2000.

4.1 Convergence Rate

Experiments show that all the algorithms are convergent. To compare their convergence rates, we plot the best individual of each generation obtained by them as shown in Fig. 8.

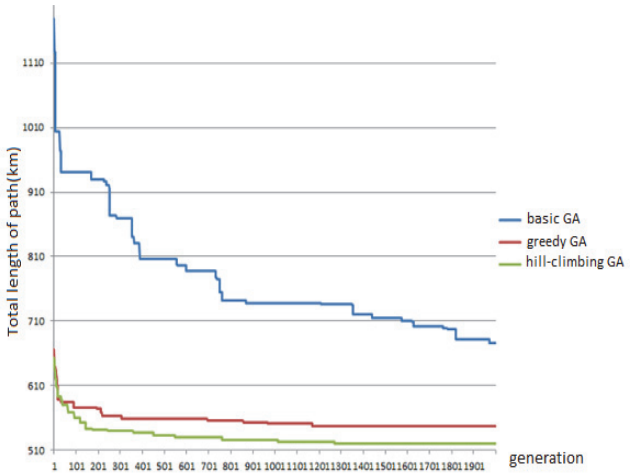

Fig. 8. Evolution performance of GAs with 2000 epochs

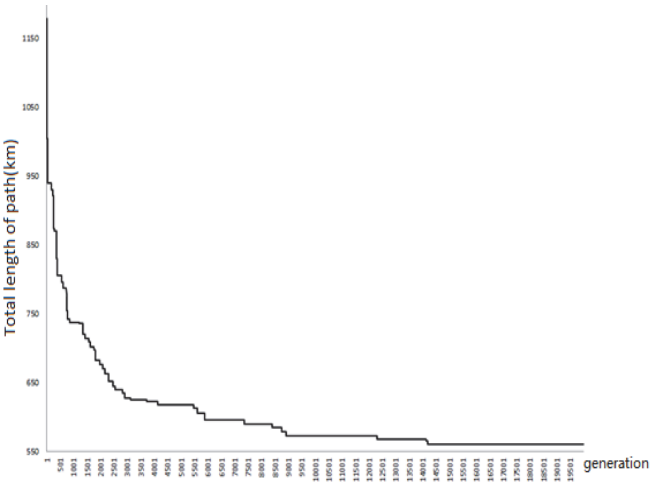

Fig. 9. Evolution performance of GA with 20000 epochs

The total path length of the best individual of the initial population of GA is about 1200km. With Greedy Algorithm, the quality of the initial population can be greatly improved. For example, Greedy GA and Hill-climbing GA converge rapidly far before their preset termination condition. However, it seems that the basic GA cannot complete its evolution within 2000 epochs. Thus, we modify the generation count of GA to be 20000. The convergence is extremely slow and the evolution ends at about the 14000th epoch and the evolution plot is shown in Fig. 9. Opposite to it, Greedy GA and Hill-climbing GA can accomplish their evolution at about the 1300th generation. This is magnitude-fold saving in computational time.

In addition, Hill-climbing GA outperforms Greedy GA. Without a hill-climbing operation, the latter needs about 1200 epochs to evolve to the result with total tour length of 550(km); while the former spends about 200 epochs only to achieve the same or better result.

4.2 Solution Quality

The results obtained in the tests are listed in Table 1.

Table 1. Results of three GAs (km)

Times	GA	Greedy GA	Hill-climbing GA
1	641.867	547.884	526.461
2	672.578	550.629	522.525
3	662.864	549.26	529.311
4	650.236	550.637	519.256
5	616.612	556.521	520.921
Mean tour length	648.831	550.986	523.695

In TABLE 1, with the same set of parameters, spending 2000 epochs, Hill-climbing GA can reach the mean tour length of 523.695(km), compared to 648.831(km) of GA and 550.986(km) of Greedy GA.

From Fig.9, we find that without the Hill-climbing operation, GA traps in the local optimum and it is hard to reach its best

result at about the 14000th epoch with the total tour length of about 560(km). With the help of the Hill-climbing operation, it is clear that after reaching the same result of Greedy GA, Hill-climbing GA can continue to obtain a better solution as shown in Fig. 8.

The total tour length of the best solution with Hill-climbing GA is 519.256(km). The solution of visit tours is:

Salesman 1: 0→35→39→3→49→45→6→5→7→2→1→

4→50→0; Salesman 2: 0→40→31→47→42→48→13→34 →9→11→14→8→12→10→38→41→0; Salesman 3: 0→32 →36→16→20→17→22→15→21→18→19→43→33→0; and Salesman 4: 0→37→44→28→24→30→26→25→27→ 23→29→46→0.

The visit routes are shown in Fig. 10.

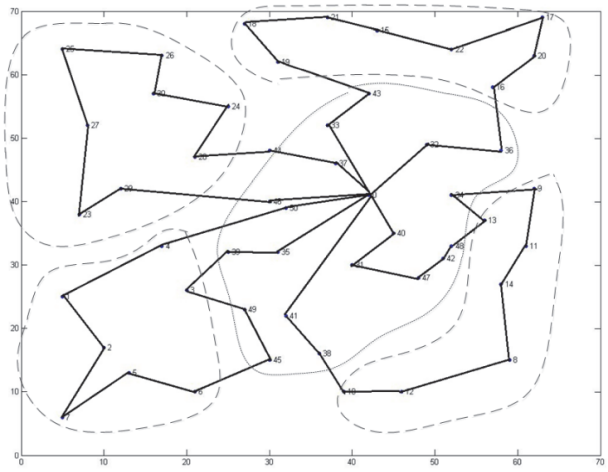

Fig. 10. Visit tours of four salesmen

In summary, (1) Hill-climbing GA overcomes the deficiency of local search of GA to some extent by keeping its global search capability. It possesses better optimization ability and yields better results than Greedy GA; and (2). The convergence rate of Hill-climbing GA is the highest among the three considered GAs. This attributes to the introduction of the hill-climbing operation in GA.

5. CONCLUSION

In this paper, we formulate a new multiple traveling salesman problem, CTSP, where different salesmen have different private city sets and share a set of public cities. It is significant for modelling the applications where multiple individuals' workspaces are not the same but partially overlap with each other. To overcome the shortcomings of the classic GA, we present two improved genetic algorithms (GA), called greedy GA and hill-climbing GA, by combining the classic one with the greedy algorithm and the hill-climbing algorithm to solve CTSP. Finally, the presented GAs are used to solve an example CTSP and the result shows that the hill-climbing GA enjoys the best performance in terms of convergence rate and solution quality. In the future, we intend to research other heuristics of CTSP and explore the related applications.

REFERENCES

Basu, A., Elnaga, A., and Al-Hajj, A. (2000). Efficient coordinated motion. *Mathematical and Computer Modelling*, vol. 31, pp. 39-53.

Bektas, T. (Jun 2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, vol. 34, pp. 209-219.

Carter, A.E., and Ragsdale, C.T. (2006). A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *European journal of operational research*, vol. 175, pp. 246-257.

Cheong, T., and White, C.C. (2012). Dynamic traveling salesman problem: Value of real-time traffic information. *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, pp. 619-630.

Gutin, G., and Punnen, A. (2002). The Traveling Salesman Problem and Its Variations. pp. 625-629. Kluwer, Dordrecht.

Hirogaki, T., Aoyama, E., Ogawa, K., Hashimoto, N., and Matsumura, M. (2005). CAM systems based on traveling salesman problem from time perspective for high density through-hole drilling. In Pts A-C (ed.), *Advances in Electronic Packaging*, pp. 1783-1789.

Johnson, O., and Liu, J. (2006). A traveling salesman approach for predicting protein functions. *Source Code for Biology and Medicine*, vol. 1, pp. 1-7.

Li, J., Sun, Q.R., Zhou, M.C., and Dai, X.Z. (Oct 13-16, 2013). A New Multiple Traveling Salesman Problem and its Genetic Algorithm-based Solution. In Proc. 2013 IEEE Int. Conf. Systems, Man and Cybernetics, pp. 1-6. Manchester, UK.

Lim, A., Rodrigues, B., and Zhang, X. (2006). A simulated annealing and hill-climbing algorithm for the traveling tournament problem. *European Journal of Operational Research*, vol. 174, pp. 1459-1478.

Ray, S.S., Bandyopadhyay, S., and Pal, S.K. (2007). Gene ordering in partitive clustering using microarray expressions. *Journal of Biosciences*, vol. 32, pp. 1019-1025.

Ryan, J.L., Bailey, T.G., Moore, J.T., and Carlton, W.B. (Dec 13-16, 1998). Reactive tabu search in unmanned aerial reconnaissance simulations. In Proc. 30th Simulation Conf. Winter. Washington, DC, vol.1, pp. 873-880.

Saleh, H.A., and Chelouah, R. (2004). The design of the global navigation satellite system surveying networks using genetic algorithms. *Engineering Applications of Artificial Intelligence*, vol. 17, pp. 111-122.

Sun, Q.R. (2011). The research and application of the colored traveling salesman problem. Master Dissertation, School of Automation, Southeast University.

Tang, L., Liu, J., Rong, A., and Yang, Z. (2000). A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex. *European Journal of Operational Research*, vol. 124, pp. 267-282,.

Toth, P., Vigo, D.(Eds.) (2002). *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia.