Solving constrained traveling salesman problems by genetic algorithms*

WU Chunguo[1,2], LIANG Yanchun[1,2**], LEE Heowpueh[2], LU Chun[2] and LIN Wuzhong[2]

(1. College of Computer Science and Technology, Jilin University; Key Laboratory for Symbol Computation and Knowledge Engineering, Ministry of Education of China, Changchun 130012, China; 2. Institute of High Performance Computing, Singapore 117528, Singapore)

Received October 13, 2003; revised November 10, 2003

Abstract Three kinds of constrained traveling salesman problems (TSP) arising from application problems, namely the open route TSP, the end-fixed TSP, and the path-constrained TSP, are proposed. The corresponding approaches based on modified genetic algorithms (GA) for solving these constrained TSPs are presented. Numerical experiments demonstrate that the algorithm for the open route TSP shows its advantages when the open route is required, the algorithm for the end-fixed TSP can deal with route optimization with constraint of fixed ends effectively, and the algorithm for the path-constraint could benefit the traffic problems where some cities cannot be visited from each other.

Keywords: constrained traveling salesman problem, genetic algorithm, Hamiltonian path, open route, fixed end.

Traveling salesman problem (TSP) is one of the most widely studied NP-hard combinatorial optimization problems, which has links with many fields of pure and applied mathematics. The TSP is to find the traveling salesman's shortest tour to pass n cities in such a way that each city is visited exactly once. There exists an implicit condition in the TSP, that is, any city can be visited from the others directly. So the issue is equivalent to finding the shortest Hamiltonian cycle within a complete simple graph. We refer to this issue satisfying the above conditions as the standard TSP. There exist many reports on solving the standard TSP using different methodologies[1,2]. The standard TSP can be used satisfactorily to solve many real prototypes in science and engineering fields. However, algorithms for standard TSP might be invalid when constraints exist, because their solutions might violate the constraints. We refer to the solution violating the constraints as an abnormal solution and the solution satisfying constraints as a feasible one. There exist various constrained TSPs in real applications, for example, the vehicle-capacity-constrained TSP and the TSP with precedence constraints (TSPPC). The first constrained TSP can be stated as: given the capacity of a vehicle, the task of the vehicle is to transport some cargos among depots. The TSPPC can be described as: there is an order imposed on the vertices of the standard TSP where the vertices should be visited and followed. These constrained TSPs can be applied to many industrial problems such as scheduling, routing decision, process sequencing, and some others[3~6]. Moon et al. in [3] proposed a genetic algorithm based on topology sort which is defined as an ordering of vertices in a directed graph. Tannenbaum in [4], Kusiak et al. in [5] and Savelsbergh et al. in [6] presented and summarized some algorithms based on dynamic programming. When these algorithms are used for solving some concrete problems, they overcome some shortcomings of the standard TSP. However, compared with the standard TSP, there is much less work on constrained TSP besides those mentioned above. In this paper we study three other kinds of constrained TSPs and present the algorithm realization schemes. When designing these algorithms for constrained TSPs, we try to make very few modifications to the standard GA-based TSP solving algorithms so that readers who are familiar with the standard TSP could use these proposed algorithms easily.

1 Traditional methods for standard TSP and their limitations

For a standard TSP with n cities, there are at

* Supported by the Science-Technology Development Project of Jilin Province of China (Grant No. 20030520) and the Key Science-Technology Project of the Education Ministry of China (Grant No. 02090)

** To whom correspondence should be addressed. E-mail: ycliang@public.cc.jl.cn (liangyc@ihpc.a-star.edu.sg)

most $n!$ Hamiltonian cycles when starting city and tour direction are taken into account. There must exist a shortest one in these finite Hamiltonian cycles. With the rapid increasing of processing speed of computers, it seems that the TSP could be solved easily with exhaustive search. However, it is impossible to implement the enumeration when the number of cities is large. Most methods based on exhaustive-enumeration would encounter a size problem with increasing dimensionality[7]. The dimension issue forces people to accept solutions obtained by heuristic or stochastic algorithms. Genetic algorithm (GA) is an excellent tool in global search, it has become the best choice to solve the TSP and researchers have developed various GA operations for the relevant problems. The GA-based algorithm for the standard TSP presented in Ref. [1] is used in this paper. For the standard TSP, the traditional GA could obtain the reasonable or the optimal solutions. Many real problems look like the standard TSP at first glance, however, they could not be solved by the traditional GA which could result in abnormal solutions. For example, drilling on a board in electronic industry expects a shorter drilling route in order to improve productivity, save power and prolong working life of tools. If the positions of the holes are arranged in a long and narrow strip it would be better than an open route implemented. In this case the traditional GA would be disadvantageous, although we could delete the longest edge in the final cycle to form an open route. It could be understood easily by referring to Figs. 1 and 2. The traditional GA would mostly generate the result shown in Fig. 1. The desirable solution for drilling problem mentioned above is shown in Fig. 2, which is unlikely to be obtained from the route shown in Fig. 1 by deleting one edge. Again, the traditional GA would fail if the 3rd and 5th vertices must be drilled first and last respectively. In this case the drilling route shown in Fig. 3 would be the most desirable one. In addition, as concerned in a traffic problem as shown in Fig. 4, there are a lake between the 1st and 2nd city, a forest between the 1st and 5th city and a private farm between the 2nd and 6th city. The traditional GA would fail to give the feasible solution shown in Fig. 4. From the above analysis, it can be seen that the study on the constrained TSP possesses both theoretical and practical significance.

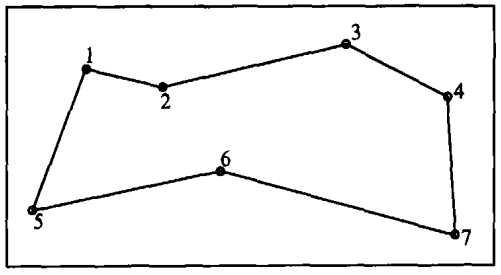

Fig. 1. Mode of the standard TSP solution.

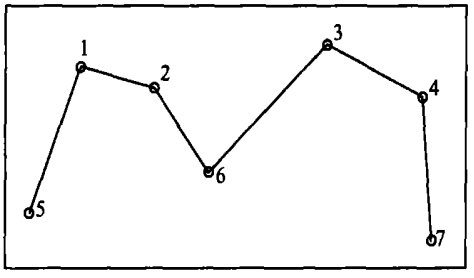

Fig. 2. Mode of an open route TSP solution.

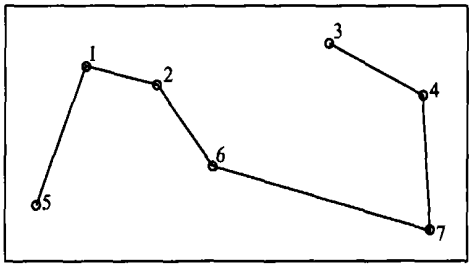

Fig. 3. Mode of the end-fixed TSP solution.

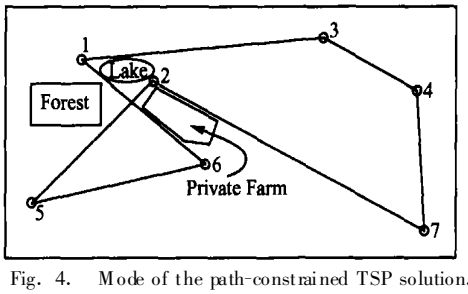

Fig. 4. Mode of the path-constrained TSP solution.

2 Some constrained TSPs and their algorithms

In this section, we consider some constrained TSP models and present the modified GA-based algorithms for solving them. These constrained TSPs can be summarized as follows:

(i) Open route TSP. It is required to search a Hamiltonian path in a complete simple graph with n vertices. For this case, the goal is to find a solution as shown in Fig. 2.

(ii) End-fixed TSP. It is required to search a Hamiltonian path with fixed starting and ending vertices in a complete simple graph with n vertices. For

this case, the goal is to find a solution as shown in Fig. 3.

(iii) Path-constrained TSP. It is required to search the shortest Hamiltonian cycle (or path) in a non-complete simple graph. Solving this model would benefit traffic problems. For this case, the goal is to find the solution as shown in Fig. 4 in which the route needs to detour to avoid the barriers such as the lake, forest and private farm.

The commonly used objective and fitness functions in GA-based algorithm for solving the standard TSP are as follows:

$$obj(S) = d(s_n, s_1) + \sum_{i=2}^{n} d(s_{i-1}, s_i), \quad (1)$$

$$f(S) = \frac{1}{obj(S)}, \quad (2)$$

where S is the individual in the GA population, s_i the ith gene in S and n the total number of cities (vertices). The three algorithms proposed in this paper for solving the constrained TSPs are:

(i) Algorithm for the open route TSP

Denote the algorithm as A1. This algorithm searches the shortest Hamiltonian path without fixed starting and ending vertices. The individual is taken as an open route, which means that the starting is from the first city and the ending is at the last city in the individual string. It differs from a cycle in the standard GA-based algorithm. For example, the individual

2 1 7 5 8 3 6 4 9

represents a visiting sequence starting from city 2 to city 1, from city 1 to city 7, …, from city 4 to city 9, and ending at city 9. The standard GA-based algorithm requires the path from city 9 to city 2 to form a complete cycle. To deal with the constrained TSP, the modified GA-based algorithm only needs to change the objective function (1) into

$$obj(S) = \sum_{i=2}^{n} d(s_{i-1}, s_i). \quad (2)$$

Performing genetic operations iteratively, the solution with the biggest fitness could approximate to the optimum solution in the feasible solution space.

(ii) Algorithm for the end-fixed TSP

Denote the algorithm as A2. This algorithm searches the shortest Hamiltonian path with fixed starting and ending vertices. The GA procedure is modified as removing the starting and ending vertices from the vertex set V, and only encoding the rest $n-2$ vertices; or as calculating the objective function using the following rule:

$$obj(S) = d(s_1, v_s) + d(v_e, s_{n-2}) + \sum_{i=2}^{n-2} d(s_{i-1}, s_i), \quad (4)$$

where v_s and v_e represent the numbers of starting and ending vertices. From the above objective function, it can be seen that although v_s and v_e do not participate in the optimization, they actually affect the fitness of each individual. So it is impossible for a "bad" individual violating fixed vertices to get a relatively high fitness. With the update of generations, the solution with the best fitness in the current population could approximate to the optimum solution in a feasible solution space. If only one vertex is fixed, either it is the starting or ending vertex, the algorithm is analogical to that in the two-end-fixed case.

(iii) Algorithm for the path-constrained TSP

Denote the algorithm as A3. Some applications may have path constraints as mentioned in Section 1, i.e. there are no paths between some city pairs. The hard measures to keep the solution feasible, for example, examining each individual and replacing it with a feasible solution if it violates constraints, would not only increase the computation work, but also harm the excellent gene pieces. Therefore, a soft measure, which overcomes these shortcomings by taking full advantage of the Darwin's evolution rule, is proposed in this paper. In addition, it could indicate the feasibility of an individual conveniently.

Firstly the distance of each city pair needs to be calculated as usual. Denote $d_{max} = \max_{1 \leqslant i, j \leqslant n} \{d_{ij}\}$. If there is not a direct path between city i and city j, then let $d_{ij} = n \times d_{max} + 1$. This modification ensures that if certain individual contains city i directly connecting city j, its objective function value is larger than $n \times d_{max}$, and its fitness function value is smaller than $\frac{1}{n \times d_{max}}$. It is obvious that under the effect of evolution rule the abnormal formal solution with fitness smaller than $\frac{1}{n \times d_{max}}$ is deserted from the current population. However, the third problem is equivalent to searching Hamiltonian cycle in a non-complete graph where the existence of the cycle is not sure. If the fitness of the best individual in the last generation is smaller than $\frac{1}{n \times d_{max}}$, there is no feasi-

ble solution in the current population.

3 Numerical experiments

To demonstrate the validities of the above algorithms, we perform the following numerical experiments. The data randomly generated are used to simulate the city locations. The positions and optimal result obtained using the standard GA are shown in Fig. 5. The parameters shown in Table 1 are used as default values. The simulations are performed on a PC with a 1600 MHz processor and 256 MB RAM.

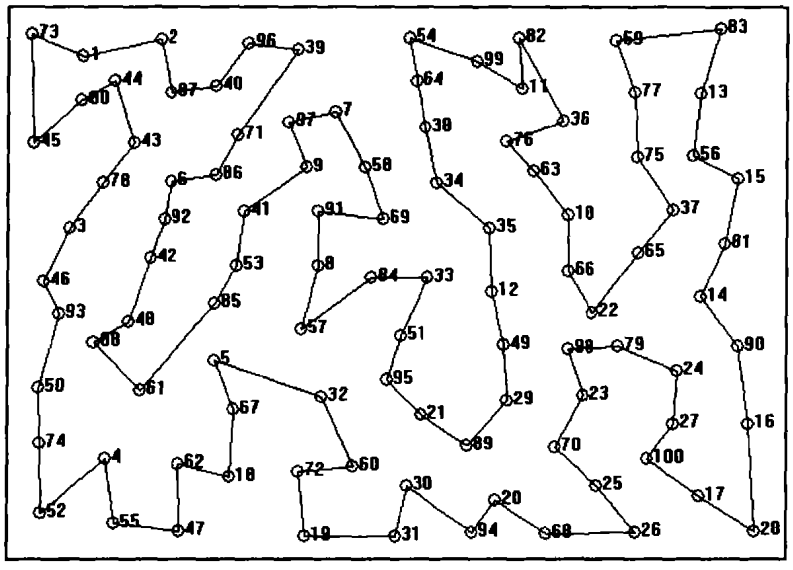

Fig. 5. Randomly generated data and optimal result using standard GA. (Initial length= 28503.82, final length= 5063.90).

Table 1. GA running parameters

Item	Crossover probability	Mutation probability	Chromosome length	Population size	Maximum generation
Parameter	0.8500	0.0005	100	100	1000

Tables 2 and 3 show the five runs for algorithms A1 and A2, respectively. The routes corresponding to the optimization results are shown in Figs. 6 and 7, respectively. As far as the best open route is concerned, the initial and optimal lengths are 22026.82 and 4630.79 by using algorithm A1, respectively; and for algorithm A2, the corresponding results are 22850.76 and 4823.57, respectively. The average reducing rates for algorithms A1 and A2 in 5 runs are 79.78% and 78.33%, respectively. In the tables the reducing rates are calculated using the following equation:

$$\text{Reducing rate} = \frac{\text{Initial length} - \text{Optimial length}}{\text{Initial length}} \times 100\%. \quad (5)$$

Numerical simulation results show that the constraint of the fixed ends restricts the search for optimal solutions, however, in some real applications the use of algorithm A2 could guarantee the feasibility of the solution. Tables 4 ~6 and Figs. 8 ~10 show the examples where the vertices are distributed in some special forms: narrow strip, regular clustering, and random clustering. Algorithm A1 is employed here to perform the optimization. The average reducing rates in Tables 4, 5 and 6 indicate that the more complex the vertices distributed, the higher reducing rates could be obtained by using algorithm A1. It indicates that the algorithm has much potential in complicated applications.

Table 2. Results for the shortest Hamiltonian path without constrained ends (A1)

	Initial length	Optimal length	Reducing rate (%)	Consumed time(s)	End vertices	Route status
1	23416.81	4684.14	80.00	1.52	22 100	Open
2	23711.15	4707.84	79.90	2.60	9 16	Open
3	23218.75	4671.39	79.88	1.46	80 100	Open
4	23104.79	4657.06	79.84	2.30	41 65	Open
5	22026.82	4630.79	78.98	1.36	93 100	Open
Average	23095.66	4670.24	79.78	1.85	—	—

Table 3. Results for Hamiltonian path with constrained ends (A2)

	Initial length	Optimal length	Reducing rate (%)	Consumed time(s)	Constrained ends	Route status
1	22361.20	4809.47	78.49	1.96	28 73	Open
2	22042.17	4880.03	78.18	1.84	28 73	Open
3	22850.76	4823.57	78.89	1.61	28 73	Open
4	21595.59	4878.69	77.41	2.09	28 73	Open
5	23096.90	4863.80	78.94	3.12	28 73	Open
Average	22389.32	4851.11	78.33	2.12	—	—

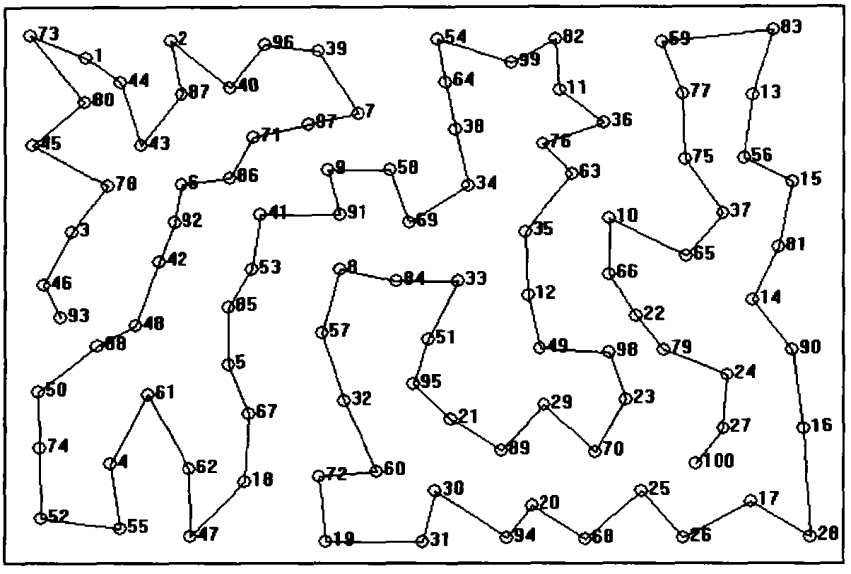

Fig. 6. Optimal Hamiltonian path without constrained vertices (A1). (Initial length= 22026.82, final length= 4630.79).

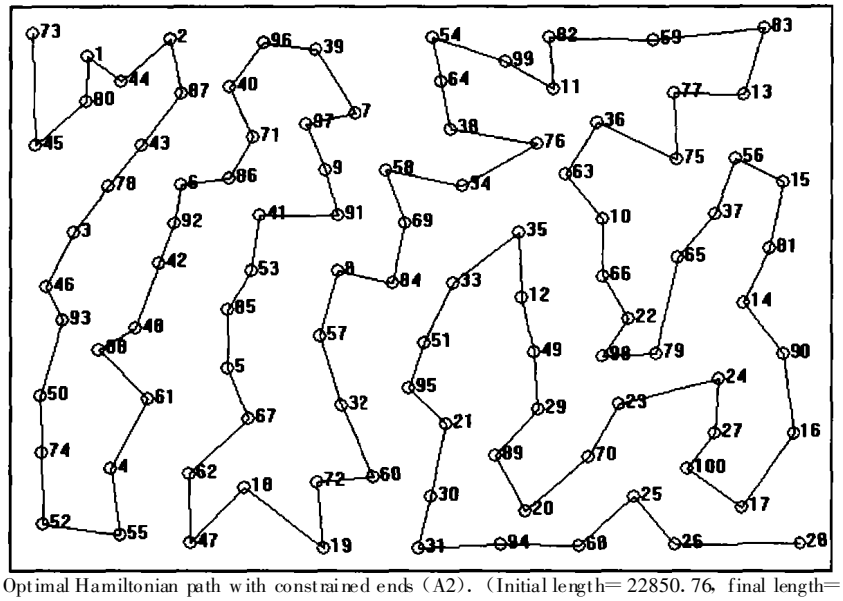

Fig. 7. Optimal Hamiltonian path with constrained ends (A2). (Initial length= 22850.76, final length= 4823.57).

Table 4. Results for Hamiltonian path in strip-like case (A1)

	Initial length	Optimal length	Reducing rate (%)	Consumed time(s)	End vertices	Route status
1	2520.37	810.24	67.85	0.78	1 18	Open
2	2359.49	810.24	67.85	0.94	1 18	Open
3	2712.14	826.85	69.51	0.69	2 18	Open
4	2371.17	810.24	65.83	0.65	1 18	Open
5	2624.65	810.24	69.13	0.95	1 18	Open
Average	2517.56	813.56	67.68	0.80	—	—

Table 5. Results for Hamiltonian path in regular clustering case (A1)

	Initial length	Optimal length	Reducing rate (%)	Consumed time(s)	End vertices	Route status
1	21312.88	3430.73	83.90	1.65	20 100	Open
2	16802.84	3457.22	83.78	1.77	16 100	Open
3	23137.34	3401.32	85.30	2.58	13 100	Open
4	21574.87	3481.42	83.86	1.63	7 100	Open
5	18870.87	3458.65	81.67	1.60	17 100	Open
Average	20339.67	3445.87	83.06	1.85	—	—

Table 6. Results for Hamiltonian path in random clustering case (A1)

	Initial length	Optimal length	Reducing rate (%)	Consumed time(s)	End vertices	Route status
1	24178.15	3637.87	84.95	1.27	3 100	Open
2	24210.97	3643.29	84.93	2.17	67 74	Open
3	24133.25	3683.33	84.74	2.71	47 12	Open
4	24885.21	3662.21	85.28	1.78	13 100	Open
5	24107.96	3704.18	84.64	1.42	31 52	Open
Average	24303.33	3666.18	84.91	1.87	—	—

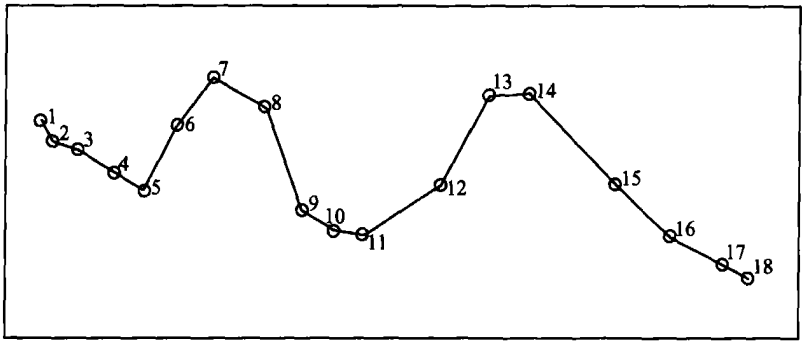

Fig. 8. Optimal Hamiltonian path for a strip case (A1). (Average initial length= 2517.56, average final length= 813.56).

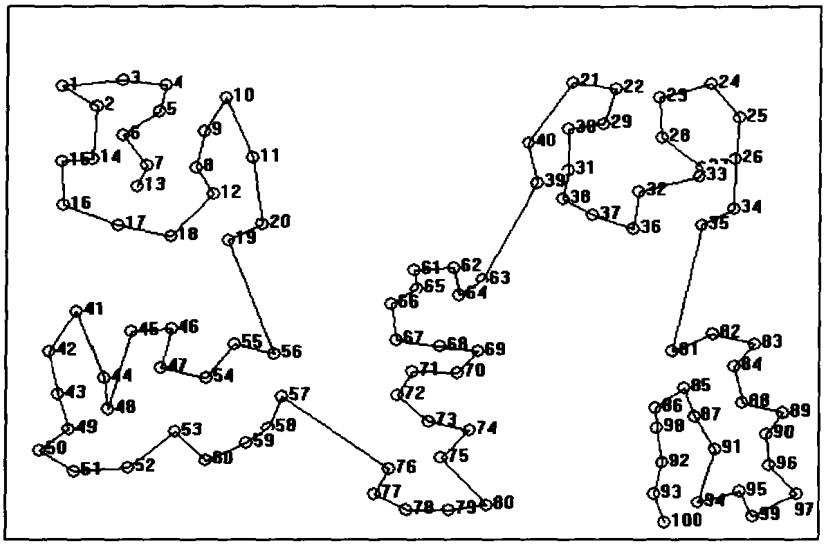

Fig. 9. Optimal Hamiltonian path for a regular clustering case (A1). (Initial length= 23137.34, final length= 3401.32).

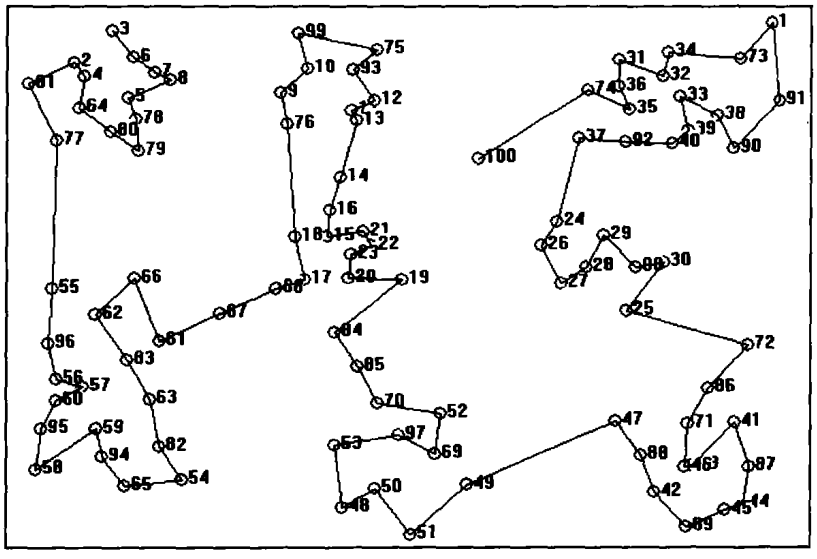

Fig. 10. Optimal Hamiltonian path for a random clustering case (A1). (Initial length= 24178.15, final length= 3637.87).

Fig. 11 shows the results obtained using algorithm A3. From the figure it can be seen that cities 16, 17, 23, 24, 25, 26, 27, 29, 70 and 79 are relatively closer to city 100. These cities are most likely connected to city 100 directly. To demonstrate the effectiveness of algorithm A3, we specify that they cannot connect city 100 directly as listed in Table 7. Fig. 11 shows that the results obtained using algorithm A3 satisfy the isolating constraints quite well.

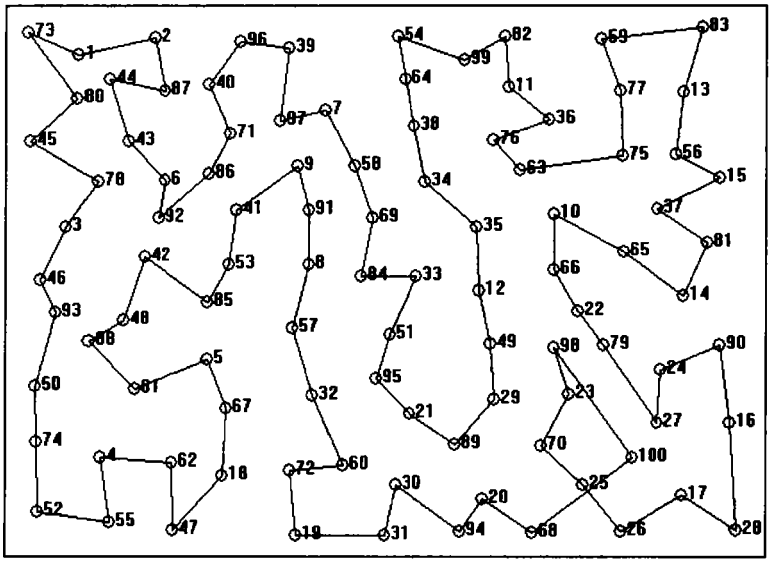

Fig. 11. Optimal Hamiltonian cycle with isolated constraints (A3). (Initial length= 26290.58, final length= 3923.45, time= 3.4 s).

Table 7. City pairs without direct path to city 100

(16, 100)	(17, 100)	(23, 100)	(24, 100)	(25, 100)
(26, 100)	(27, 100)	(29, 100)	(70, 100)	(79, 100)

4 Conclusions and discussions

This paper studies three kinds of constrained TSPs which cannot be solved by making use of the traditional GA for standard TSP. The objective function is designed to take full advantage in the process of algorithm design, as it determines the individual fitness according to the rule in the fitness calculation. Simulations show that, among these proposed algorithms, algorithm A1 could provide a better solution if the open route is required and there are no fixed ends, algorithm A2 could deal with route with prefixed ends effectively, and Algorithm A3 could give an effective solution when either an open or closed route is required in an incomplete graph, which would benefit the traffic problems in which there are no direct paths between some cities.

References

1 Liang, Y. C. et al. Solving traveling salesman problems by genetic algorithms. Progress in Natural Science, 2003, 13(2): 135.
2 Katayama, K. et al. The efficiency of hybrid mutation genetic algorithm for the travelling salesman problem. Mathematical and Computer Modeling, 2000, 31(10~12): 197.
3 Moon, C. et al. An efficient generic algorithm for the traveling salesman problem with precedence constraints. European Journal of Operational Research, 2002, 140(3): 606.
4 Tannenbaum, P. et al. Excursions in Modern Mathematics (4th ed.), NJ: Prentice Hall, 2000.
5 Kusiak, A. et al. Modeling and solving the flexible forging module scheduling problem. Engineering Optimization, 1987, 12(1): 1.
6 Savelsbergh, M. W. P. et al. The general pickup and delivery problem. Transportation Science, 1995, 29(1): 17.
7 Mingozzi, A. et al. Dynamic programming strategies for the TSP with time windows and precedence constraints. Operations Research, 1997, 45(3): 365.