ORIGINAL RESEARCH PAPER

Computer Science

AN IMPROVED GENETIC ALGORITHM FOR SOLVING TRAVELING SALESMAN PROBLEM

KEY WORDS: Multiple heuristic techniques ; Optimize the problem ; genetic algorithm

Yan Wanjie Tianjin Polytechnic University

Wan Zhenkai* Tianjin Polytechnic University *Corresponding Author

ABSTRACT

It is very important to solve the traveling salesman problem accurately in many fields of scientific research, such as spaceflight, satellite and so on. By studying the traveling salesman optimization problem, several heuristic techniques are adopted to reduce the mutation rate and crossover rate with time, and the adaptive algorithm is used to calculate the average population fitness and the current optimal fitness of the population and to update its parameters. The algorithm is improved by carefully designing application program, introducing elite operator, eliminating operator and terminating checking method. Twenty data sets are extensively experimented and compared with four existing methods. The results show that the improved genetic algorithm can significantly improve the accuracy and convergence of genetic algorithm and effectively avoid the problem of local optimal solution. And can achieve two orders of magnitude acceleration effect

INTRODUCTION

Traveling Salesman Problem (TSP), which can be traced back to the 19th century, is still unresolved today. The traveling salesman problem is the optimal solution for traversing a set of cities. It is a typical optimization problem, which requires each city to be visited and must be visited once, and can be applied to many other scenarios. For example, the handshakes of national leaders at international conferences, etc., for some applications, the concept of city can correspond to customers, or even the welding points of microchips. Therefore, how to solve TSP problem quickly and effectively has high practical value[1].

In its simplest form, a city can be represented as a node on the graph, and the distance between each city can be represented as the length of the edge as shown in figure 1. A "route" or "route" defines the edges of the application and the order in which it is used. Then the route score is calculated by accumulating the edges of the route.

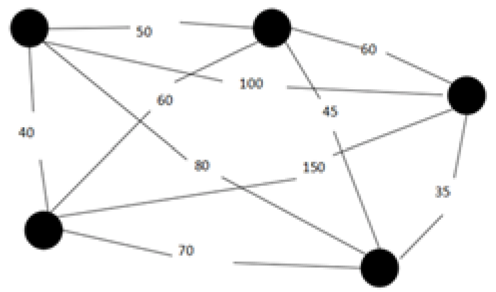


Figure 1. The city and the distance between them

In graph theory terms, for n cities v = {v1, v2, ……, vn }The order of access is t = (t1, t2, …... ti, …,tn), ti v, (i = 1,2,……, n), And remember tn+1 = t1.

Then its mathematical model is:
$\min l = a^{d}(t(l), t(i+1)),(i = 1,2,\ldots, n)$

If there are only a few cities, it is easy to find the optimal solution with violence algorithm, but as the number of cities increases, they become more and more challenging. In this paper, the traveling salesman optimization problem is analyzed and studied, and several heuristic techniques are used to reduce the mutation rate and crossover rate over time. The adaptive algorithm is used to calculate the average population fitness and the current optimal fitness of the population and to update its parameters. By using the multi-core characteristic of the computer, the elite operator is introduced and the elimination operator is introduced by carefully designing the application program. The optimization ability is greatly improved by the method of terminating inspection. The results show that the improved genetic algorithm can improve the accuracy and convergence of the genetic algorithm, and effectively avoid the problem of local optimal solution, and even increase the population size over time. It still has a very good ability to find the best.

BACKGROUND

n cities, each city must be visited and can only be accessed once, and need to find a shortest access route containing all n cities[9]. For this problem, when calculating the distance between two cities, the shortest distance between cities is used, and the calculation distance is as follows:

$$D_{ab} = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$$

Biological evolution is the process of natural selection, which was first proposed by Darwin in 1859 in his book The Origin of Species. A chromosome is often referred to as a candidate solution in a genetic algorithm because the genetic algorithm uses a chromosome to encode a candidate solution. The various possible settings for a particular trait are referred to as "alleles", the positions in which the traits are encoded in the chromosome are referred to as "gene numbers" and the particular genome is referred to as "genotypes". Genetic algorithms reveal the process of survival of the fittest and population evolution.

SOLUTION PRINCIPLE

The fitness function is a special objective function that can be used to describe whether a given target can converge a given solution and find the optimal solution. The fitness function is usually used in genetic algorithms to test an individual's viability and population fitness.

The flow chart using the genetic algorithm to solve the problem is shown in Figure 2.

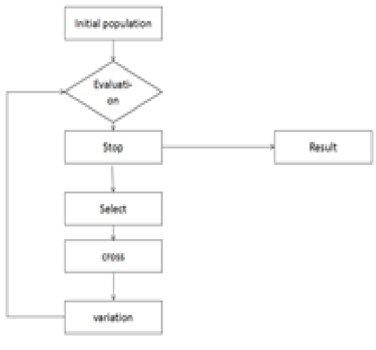


Figure 2 Genetic algorithm solving problem flow chart

Volume-7 | Issue-12 | December-2018 | PRINT ISSN No 2250-1991

The pseudo code of the algorithm is shown in Table 1:

Table 1 Basic genetic algorithm pseudo code

Traditional genetic algorithm
1:genaration = 0;
2:population[genaration] = initializePopulation(populationSize);
3: evaluatePopulation(population[genaration]);
4**: While** isTerminationConditionMet() == false **do**
5: parents selectParent(population[genaration]);
6: population[generation+1] = crossover(parents);
7: population[generation+1] = mutation (population[genaration+1]);
8: evaluatePopulation(population[genaration]);
9: genaration++;
10**: End loop;**

The pseudocode begins with the creation of the initial population of the genetic algorithm, and then the population is evaluated to determine the fitness value of the individual. Next, check to see if the termination condition of the genetic algorithm has been met, and if not, the genetic algorithm begins to loop. After the first round of selection, crossover and mutation, the population is re-evaluated, and thereafter, selection continues.

It is then referenced in the chromosome in the order of the candidate routes. This type of coding uses the order of the genes, called permutation coding.

Initialization: Generate some random cities, using x, y coordinates to uniquely identify the city location.

Population evolution: Using the elite individual preservation strategy, the mathematical model is as follows:

$$P_i = \text{fitness}(i) \;/\; \sum_{i=1}^{\text{chromosome}} \text{fitness}(i),\; Q(i) = \sum_{j=1}^{i} P_i$$

Corresponding mutation operations are performed on individual individuals, usually using reverse mutation operators. After performing related operations such as retention, elimination, selection, crossover, and mutation, re-evaluate the population and individual. If the evaluation result meets the requirements, the algorithm ends. Otherwise, go to step 4 of Table 1 for the next round of evolutionary operations.

IMPROVED ALGORITHM

Heuristic crossover operator:

The crossover operator design patterns used in the traditional genetic algorithm to solve TSP are single point intersection, two point intersection, sort intersection, sequence intersection, reverse order intersection and so on. However, the following two problems occur when using the crossover operator. (i) When the population evolves to suboptimal over time, the crossover rate and the mutation rate still retain higher values, resulting in a much lower probability of population degradation; (ii) It is easy to fall into the local optimal solution, so that the true optimal solution cannot be found.

The two most advanced heuristic methods popular in genetic algorithms are simulated annealing in [12] and tabu search in [13]. However, these two methods still cannot solve the above problems. In this paper, a new heuristic crossover operator is introduced in combination with literature [12] and [13]. This operator combines differential operator and information entropy to obtain better evolutionary progeny. By traversing the chromosomes of all individuals in the population, find the two genes with the shortest path as the two genes of the offspring, and then compare the next gene in the parent with the shortest distance from the two cities as the offspring, and so on. Until the entire city is traversed.

Fractional calculus:

Fractional calculus is a branch of mathematical analysis. It mainly studies how to define real power or complex power for differential operator D and integral operator J. The general representation of the differential operator D and the integral operator J is as follows:

$$Df(x) = \frac{d}{dx}f(x),\quad \mathcal{J}f(x) = \int_0^x f(s)ds$$

Information entropy:

The calculation of information entropy is very complicated, and its calculation formula is as follows:

$$H(x) = E[I(xi)] = E\left[\log\left(2,\ \frac{1}{p(xi)}\right)\right] = -\sum p(xi)\log\left(2,\ p(xi)\right)\ (i = 1,2,\ldots n)$$

Elimination of operators: In order to make the evolved offspring population more excellent, this paper introduces the elimination operator as e, in order to eliminate the inferior genes in the contemporary population, so that the genetic algorithm can be improved as much as possible, so as to improve Excellent gene of the offspring. First, the populations are sorted according to the fitness from high to low, and then the individuals at the end, that is, the inferior genes, are removed from the group according to the size of the operator. The new population obtained will be the better individuals after screening. At this time, the basic operation of the genetic algorithm of the population, selection, crossover, and mutation will result in better progeny populations.

Adaptive adjustment mechanism

The individual fitness and population fitness values of the evolutionary population are calculated by a certain function. The fitness function used by the improved algorithm takes the crossover rate, the mutation rate, and the elite operator as the main optimization objects, and adds the elimination operator, which enables it to measure the optimal region of the search space, and centrally search and carry out the corresponding inheritance. operating.

The process of calculating the fitness value by the fitness function can be expressed as:

$$\text{fit}(x) = \lambda \frac{\text{fit}(x) - \text{fit}_{\min}}{\text{fit}_{\max} - \text{fit}_{\min}} + (1-\lambda)\frac{\|\nabla f(x) - \nabla f_{\min}\|}{\|\nabla f_{\max} - \nabla f_{\min}\|}$$

$\nabla f(x)$ Is the gradient of the objective function f(x) at x, ∇f_{max} and ∇f_{min} They are defined as follows:

$$\nabla f\min = \left(\min\left\{\frac{\partial f(v_1)}{\partial (v_1)_1},\ \ldots,\ \frac{\partial f(v_{pt})}{\partial (v_{pt})_1}\right\},\ \ldots,\ \min\left\{\frac{\partial f(v_1)}{\partial (v_1)_n},\ \ldots,\ \frac{\partial f(v_{pt})}{\partial (v_{pt})_n}\right\}\right),$$

$$\nabla f\max = \left(\max\left\{\frac{\partial f(v_1)}{\partial (v_1)_1},\ \ldots,\ \frac{\partial f(v_{pt})}{\partial (v_{pt})_1}\right\},\ \ldots,\ \max\left\{\frac{\partial f(v_1)}{\partial (v_1)_n},\ \ldots,\ \frac{\partial f(v_{pt})}{\partial (v_{pt})_n}\right\}\right)$$

one of them l [0,1], It is called the control factor to reflect the importance of the fitness function and the rate of change of the function to solve the problem. fit_{max} and fit_{min} Representing the highest fitness and minimum fitness of the evolutionary population, respectively. The fitness function can improve over time, and if the population is still improving rapidly, the algorithm will continue. Once the population stops improving, the fitness function will notify the algorithm to end. The improvement of the fitness function over time is mainly by measuring the continuous generation of the best individual without improvement. If the continuous generation of no improvement has exceeded a certain threshold, for example, the generation does not improve, the algorithm can be stopped.

Performance improvement

In this improved algorithm, the fitness function is the most computationally intensive part, so it makes sense to improve the code of the fitness function in order to get the best performance return. When calculating the distance between two cities, use the shortest distance between cities

$$D_{ab} = \sqrt{(X_a - X_b)^2 + (Y_a - Y_b)^2}$$

x_a, x_b representative city A B X-axis seat, The same reason y_a, y_b representative city A B Y-axis seat, Dab representative the shortest path of city A and B.

Modern computers are generally multi-core processors. Using this feature of the computer, parallel operations are performed at the fitness function code block, so that the fitness function runs on multiple cores of the machine, and the multiple cores of the computer share the workload. Will greatly improve the performance of the algorithm.

EXPERIMENT ANALYSIS

In order to analyze the performance of the improved algorithm, the experiment is compared with the existing algorithm, and the performance of the improved algorithm is analyzed from the optimal solution and time.

The specific analysis is shown in Table 2 to Table 6 below.

These six tables record the specific running time and the optimal solution value of the number of cities of different sizes under the five algorithms. By analyzing Table 1, in the case of a small number of cities, the optimal solutions obtained by the five algorithms are not too different. The improved algorithm is relatively small, although the optimal solution is similar to the other four methods, but The optimal solution is better than the other four methods, and it is obvious from its time that it runs faster than other methods. With the increasing number of cities, the number of cities in Table 3 has increased from 300 to 1,500. In the case of 1500 cities, the running time under the traditional genetic algorithm is 4379 ms. The other three algorithms have an annealing, greed and taboo of 4001 ms respectively. 2801ms, 3987, although the speed has improved, their running time is still too long. The improved algorithm introduces the elite operator, eliminates the operator, terminates the checking method, adopts parallel coding during programming, and gives full play to the multi-core characteristics of the computer. The running time is 292ms. It can be seen that the improved algorithm has a great improvement in performance, which is better than the existing four methods. Some people may say that our experimental city is not large enough, then our next experiment, from Table 4, Table 5, the city scale continues to increase, from 3,000 cities to 20,000 cities, 50,000 cities to 500,000 cities, 5 species The optimal solution of the algorithm is getting better and better, and the running time is increasing. When the scale of 500,000 cities, the running time of the traditional genetic algorithm is nearly half an hour, the annealing algorithm and the tabu algorithm are all in ten minutes. The greedy algorithm is in about five minutes. Let's look at the improved algorithm, which runs at 1.65 minutes. Let us analyze from the perspective of the optimal solution. The improved algorithm has an optimal solution of 613, while the other four algorithms are around 700. The data records from Table 5 of Table 4 indicate that the improved algorithm is still superior to other existing methods in the case of large city scales.

The curves in Figure 3 and Figure 4 are from top to bottom: traditional algorithm, annealing algorithm, greedy algorithm, tabu algorithm, improved algorithm.

Figure 3 and Figure 4, it can be clearly seen from Figure 3 that as the size of the city continues to increase, the existing four algorithms are more or less degraded, except for the optimal solution. The improved algorithm is good at avoiding this limitation, and the optimal solution is better than them. Analysis of Figure 4, when the city scale reaches tens of thousands, even after millions, ten million, the running time of the existing algorithm is greatly increased, especially the traditional algorithm, its running time even reaches 27 hours, which can be seen in the city scale. In the case, they take too long, the performance is too poor, and not stable enough, and the improved algorithm is better than the existing algorithm, whether it is time-consuming or stable. From Figure 4 above, it can be clearly seen that the improved algorithm has a slower slope and has good stability. After re-analysing Figure 3, the slope of the improved algorithm is larger than the others, which proves that it has the characteristics of fast convergence.

By analyzing Figure 3, Figure 4, and the above five tables, we can conclude that the improved algorithm is superior to existing algorithms in both convergence and performance.

CONCLUSION

This paper designs an improved high performance genetic algorithm to solve the TSP problem. Introducing elite operators, eliminating operators, initializing population methods and implementing multiple heuristic crossover operators, the optimization ability is significantly improved, so that future generations evolve better population quality, and the multi-core characteristics of computers are utilized to make full use of modern computers. High concurrent computing power. We conducted 20 sets of experiments. Compared with other existing algorithms (traditional algorithms, taboo algorithms, annealing algorithms, greedy algorithms), we proved that the proposed algorithm is based on both convergence and optimization performance. Better than the existing calculations

Table 2 Comparative analysis of experimental results

	10 cities		30 cities		50 cities		100 cities	
algorithm	optimal solution	time/ms	optimal solution	time /ms	optimal solution	time /ms	optimal solution	time /ms
traditional genetic algorithm	3899	297	3476	522	3224	724	2860	903
simulated annealing algorithm	3899	297	3476	522	3224	724	2860	903
Greedy algorithm	3899	260	3476	509	3109	698	2798	789
Tabu search algorithm	3899	297	3476	519	3198	705	2854	900
Improved genetic algorithm	3880	110	3200	213	3021	223	2592	229

Table 3 Comparative analysis of experimental results

	300 cities		500 cities		900 cities		1500 cities	
algorithm	optimal solution	time /ms	optimal solution	time /ms	optimal solution	time /ms	optimal solution	time /ms
traditional genetic algorithm	2283	1664	2207	1891	1725	2993	1669	4379
simulated annealing algorithm	2279	1597	2103	1698	1714	2654	1598	4001
Greedy algorithm	2156	1345	1979	1501	1684	2139	1403	2801
Tabu search algorithm	2206	1546	2016	1647	1701	2567	1576	3987
Improved genetic algorithm	1845	246	1798	264	1546	278	1297	292

Table 4 Comparative analysis of experimental results

	3000 cities		6000 cities		12000 cities		20000 cities	
algorithm	optimal solution	time /s	optimal solution	time /s	optimal solution	time /s	optimal solution	time /s
traditional genetic algorithm	1422	8.850	1224	17.238	1151	17.403	1016	55.338
simulated annealing algorithm	1359	7.36	1129	8.36	1099	10.36	1001	39.98
Greedy algorithm	1249	5.46	986	6.32	974	6.79	951	19.87
Tabu search algorithm	1343	6.36	1098	7.39	1029	8.19	985	29.97
Improved genetic algorithm	1021	0.4	945	0.68	901	2	872	3.75

Table 5 Comparative analysis of experimental results

	50000 cities		100000 cities		200000 cities		500000 cities	
algorithm	optimal solution	time /m	optimal solution	time /m	optimal solution	time /m	optimal solution	time /m
traditional genetic algorithm	925	2.25	854	4.57	850	11.68	785	28.92
simulated annealing algorithm	906	1.43	823	2.16	812	4.112	774	10.74
Greedy algorithm	847	0.87	801.21	1.02	779	2.31	751	5.23
Tabu search algorithm	894	1.28	816	2.01	789	4.101	761	9.64
Improved genetic algorithm	759	0.13	687	0.36	659	0.67	613	1.65

Table 6 Comparative analysis of experimental results

	1000000 cities		3000000 cities		5000000 cities		10000000 cities	
algorithm	optimal solution	time /h	optimal solution	time /h	optimal solution	time /h	optimal solution	time /h
traditional genetic algorithm	814	0.98	847.03	6.14	967.35	17.24	898	27.33
simulated annealing algorithm	759	0.53	743	1.32	732.01	4.65	730.79	10.04
Greedy algorithm	719	0.19	671	0.42	629.7	1.01	628.19	1.88
Tabu search algorithm	747	0.41	729	0.84	724	2.31	723.56	6.98
Improved genetic algorithm	412.12	0.066	382.47	0.11	380.98	0.13	379.89	0.17

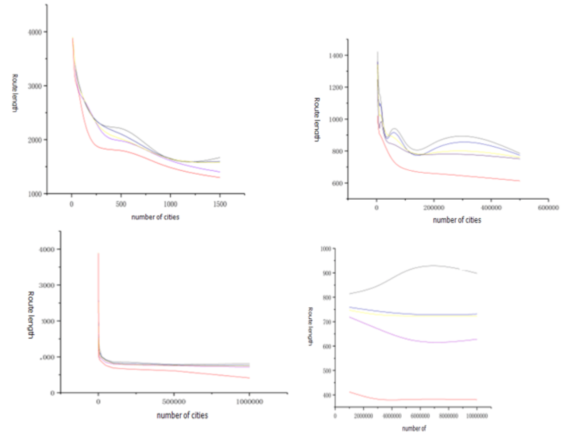
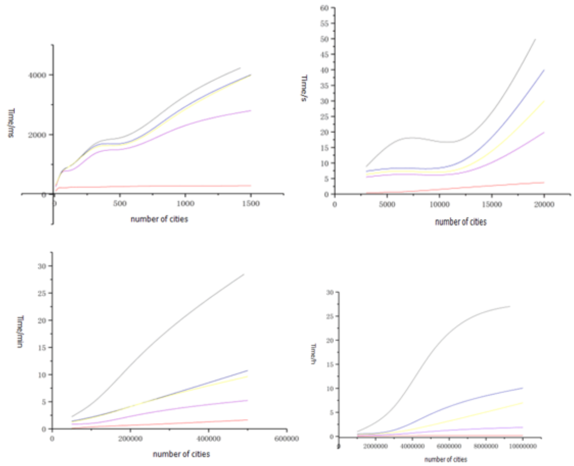


Figure 3 algorithm optimal solution curve comparison



REFERENCES:

[1] YU Ying-ying,CHEN Yan,LI Tao-ying.Improved genetic algorithm for traveling salesman problem[J].Control and Decision,2014,29(08):1483-1488.

[2] Liu Hehua,Cui Chao,Chen Jing.An Improved Genetic Algorithm for Solving Traveling Salesman Problem[J].Journal of Beijing Institute of Technology,2013,33(04):390-393.

[3] YAN Junzhong, HUANG Zhen, LIU Xiaonian. An Ant Colony Algorithm for Solving Traveling Salesman Problem[J]. Journal of Computer Research and Development, 2009, 46(6): 968-978.

[4] Li Shuai,Dang Xin,Wang Xu,Wu Jigang.Update Strategy and Algorithm in Replica Placement[J].Computer Science and Exploration,2016,10(11):1633-1640.

[5] WANG Jin-hai,DU Zheng-gao,ZHENG Yu,KONG Ying,HONG Hui,QIU Qian.Effects of low-intensity power frequency magnetic field on the distribution characteristics of sEPSC in hippocampus of brain slices[J].Journal of Tianjin Polytechnic University,2015,34(04):52- 57.

[6] NIE Jingyun, LI Chunqing, LI Weiwei, WANG Wei.Study on the Least Squares Support Vector Machine Optimized by Genetic Algorithm in MBR Simulation Prediction[J]. Software, 2015, 36(05): 40-44+48.

[7] Liu Chaohua,Zhang Yingjie,Zhang Wei,Wu Jianhui.The Fusion of Ant Colony Algorithm and Immune Algorithm and Its Application in TSP[J].Control and Decision,2010,25(05):695-700+705.

[8] Yannis Marinakis, Magdalene Marinaki. A hybrid multi- swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem[J]. Computers and Operations Research, 2010, 37(3): 432-442.

[9] Darrell Whitley, Doug Hains, Adele Howe. A hybrid genetic algorithm for the traveling salesman problem using generalized partition crossover[C]. Proc of the 11th Int Conf on Parallel Problem Solving from Nature. Berlin: Springer Heidelberg, 2010, 6283: 566-575.

[10] Zakir H Ahmed. Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator[J]. Int J of Biometrics and Bioinformatics, 2010, 3(6): 96-105.

[11] Murat Albayrak, Novruz Allahverdi. Development a new mutation operator to solve the traveling salesman problemby aid of genetic algorithms[J]. Expert Systems with Applications, 2011, 38(3): 1313-1320.

[12] Lin W, Delgadofiras Y G, Gause D C, et al. Hybrid Newton-Raphson genetic algorithm for the travelling salesman problem[J]. Cybernetics and Systems, 1995, 26(4): 387-412.

[13] Mu Ahua, Zhou Shaolei, Yu Xiaoli. A Fast Adaptive Genetic Algorithm and Its Simulation Research[J]. Journal of System Simulation, 2004, 10(1): 122-125.

[14] Su Nan,Lu Jing,Wang Dongliang.Optimization of Vehicle Routing Problem Based on Genetic Algorithm[J].Automobile Technology,2016(06):4-6.

[15] Zhang Yanxiang,Yu Yuxian.Improved Genetic Simulated Annealing Algorithm for Solving TSP[J]. Intelligent computers and applications,2017,7(03):52-54.

[16] GUO Ye-cai,ZHANG Jie-ru,ZHANG Bing-long.A Wavelet Blind Equalization Algorithm for Double-stranded DNA Computation Based on Tabu Search[J].Journal of System Simulation,2017,29(01):21-26.

[17] Zhou Cong,Zheng Jinhua.An Improved TSP Heuristic Crossover Operator[J]. Computer Engineering and Applications,2008(09):37-39+54.
[18] Ren Ziwu, Yu Yee. Improvement of Adaptive Genetic Algorithm and Its Application in System Identification[J]. Journal of System Simulation, 2006, 18(1): 41-43.
[19] Deng Xianxi. Research and Improvement of Genetic Algorithm for Solving TSP Problem [D]. Shenyang: School of Information Science and Engineering, Northeastern University, 2008.
[20] Srinivas M, Patnail K L M. Adaptive probabilities ofcrossover and mutation in genetic algorithms[J]. IEEETrans on System, Man and Cybernetics, 1994, 24(4): 656-667.
[21] Ren Haiyan, Chen Feixiang. Improvement of adaptive genetic algorithm and its application in curve simplification[J]. Computer Engineering and Applications, 2012, 48(11): 152-155.
[22] Xiang Zuoyong, Liu Zhengcai, Shen Pingan. An Improved Adaptive Genetic Algorithm Based on Evolutionary Stage[J]. Journal of Wuhan University: Engineering, 2008, 41(1): 133-136.
[23] Peng Danping, Lin Zhiyi, Wang Jiangqing. An improved genetic algorithm for solving TSP[J]. Computer Engineering and Applications, 2006, 42(13): 91-93.