An improved genetic algorithm based on k-means clustering for solving traveling salesman problem

Conference Paper · December 2016

DOI: 10.1142/9789813200449_0042

4 authors, including:

Lizhuang Tan
Qilu University of Technology (Shandong Academy of Sciences)
24 PUBLICATIONS **55** CITATIONS

SEE PROFILE

An improved genetic algorithm based on k-means clustering for solving traveling salesman problem*

Li-Zhuang Tan, Yan-Yan Tan†, Guo-Xiao Yun and Chao Zhang
School of Information Science and Engineering, Shandong Normal University, Jinan, 250014, China
Shandong Provincial Key Laboratory for Novel Distributed Computer Software Technology, Shandong Normal University, Jinan 250014, China
†*E-mail: yytan928@163.com*
www.sdnu.edu.cn

Based on the k-means clustering, we design and implement an improved genetic algorithm(IGA) to find the best solution of traveling salesman problem. Comparing with classical genetic algorithm(CGA), IGA includes three characteristics. Firstly, k-means clustering method is used on all the cities to reduce the complexity of the problem, through which the cities are divided into several groups. Secondly, genetic algorithm is adopted in each group for optimizing the sub-path. Thirdly, evolution operation is used on the inter-group for integral optimization. At last, we trial run a great quantity of experiments to test the performance of IGA in this paper. The experimental results show IGA is more effective than CGA, especially for large-scale traveling salesman problems.

Keywords: Traveling Salesman Problem; Genetic Algorithm; K-means; Hybrid Mutation.

Introduction

Traveling Salesman Problem(TSP) is a classical combination optimization problem. The TSP was first formulated as a mathematical problem in 1930 and become increasingly popular after 1950[1]. Different types of TSP problems had been studied by the researchers during the recent years[2-7]. TSP has many real world applications in planning scheduling, VLSI routing problems, etc. In TSP, the salesman must be given a closed shortest tour between n cities.

G=(V,E) is a complete undirected graph with a vertex set V={1,2,…,n}, an edge set E. In this paper, we consider the distance from city i to city j is the same

*This work is supported by the Natural Science Foundation of China (No.61401260),Shandong Province Young and Middle-Aged Scientists Research Awards Fund (No.BS2014DX006), and Shandong Normal University Training Programs of Innovation and Entrepreneurship for Undergraduates (No.201501259).
†Corresponding author address: 88 East Wenhua Road, Jinan, Shandong P. R. China.

as from city j to city i. We define $d(c_i, c_{i+1})$ as the distance traveled between cities. The objective function is as Eq.(1)[3]:

$$\min D = (\sum_{k=1}^{n-1} d(c_k, c_{k+1})) + d(c_n, c_1) \quad (1)$$

where c_k presents kth and c_1 is first and c_n the last city on the tour. If the coordinates $(c_i x, c_i y)$ of all cities (c_i) that the salesman will visit is known, we can rewrite Eq.(1) as Eq.(2)[4]:

$$\min D = \sum_{i=1}^{n-1} (\sqrt{(c_i x - c_{i+1} x)^2 + (c_i y - c_{i+1} y)^2} + \sqrt{(c_n x - c_i x)^2 + (c_n y - c_i y)^2}) \quad (2)$$

The exact algorithms can solve the TSP by a number of steps. But with the scale of TSP becomes large, for example,100 cities with approximately 10155 different solutions. Consequently, many bioinspired algorithms are carried out to obtain accepted solutions for NP-hard problems with short running time, such as ant colony optimization, simulated annealing, genetic algorithm and so on.

Genetic algorithm is an efficacious technique based on natural election for problems with huge search, such as TSP, in which the initial population decides iterations. The aim of GA is to achieve an approximate solution in a large-scale problem thought a couple of steps like selection, crossover, and mutation. Compared with other standard search algorithms, its advantages mainly consist in that the search is conducted using information of a population of tours instead of just one tour[6-8]. Aside from the foregoing content, the GA evaluates the unfit quality of the individual by the numerical value of fitness function, reduces the risk of being immersed in a local optimum when using heuristic algorithms.

K-means algorithm is one of the major algorithms of clustering analysis and a kind of clustering algorithm based on partitioning. The algorithm randomly chooses *k* points as the initial clustering centers. In order to reduce the complexity of TSP, especially for large-scale problems, we first use the *k*-means method to cluster cities into *k* groups. Then, genetic algorithms are carried out both in intra-group and inter-group for optimizing paths. An improved genetic algorithm(IGA) based on *k*-means clustering is proposed in this paper. In order to test the performance of our proposed IGA, a great quantity of experiments are implemented on IGA, and the experimental results are compared with those obtained by classical genetic algorithm(CGA) both on accuracy and running efficiency.

The remainder of this paper is organized as follows. Section 2 describes the *k*-means clustering algorithm used in this paper. Section 3 presents the genetic operations adopted in our proposed algorithm. Then in Section 4, experiments, comparing results and discussions are given. Finally, we conclude this paper in Section 5.

1. Clustering Method

The clustering method used in this paper is to convert a large-scale TSP into several small-scale problems by *k*-means. Its main idea is dividing *n* cities into several groups, and using genetic algorithm in each groups to find the shortest sub-traveling route. The use of the clustering is to accelerate the convergence and improve the accuracy of genetic algorithm.

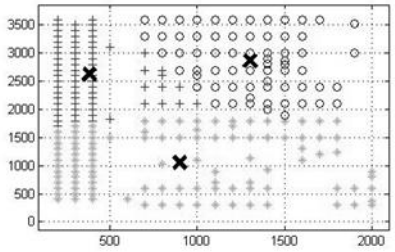

Fig. 1. *k*-means clustering

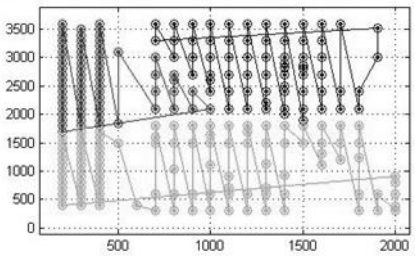

Fig. 2. Example result of clustering formation.

The first step of the clustering algorithm is to form *k* groups by *k*-means clustering with $k = [\sqrt{n} + 0.5]$, each groups is a cluster. We can number the cluster from 1 to *k*.

The second step is to find the shortest loop for the given vertexes, which divided into groups with genetic algorithm. Shown as Figure. 2.

The third step is to choose an edge of the shortest loop randomly in each cluster, and deleted the edge and one of the two adjacent vertexes will be the

start vertex and other will be the end. Then rewire the start and the end one by one. Shown as Figure. 3.

The fourth step is to repeat the third step so as to generate the initial population.

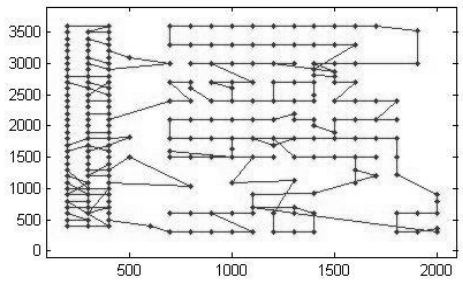

Fig. 3. The initial population.

2. Genetic Evolution

In the lower gradation, the genetic algorithm searches the shortest loop for each cluster. In the higher gradation, we delete the chosen edge from the above loop.

3.1. *Intra-group evolution operation*

Intra-group evolution operation, which applied cluster by cluster, is to find the shortest loop. We use genetic algorithm to optimize the loop during the operation.

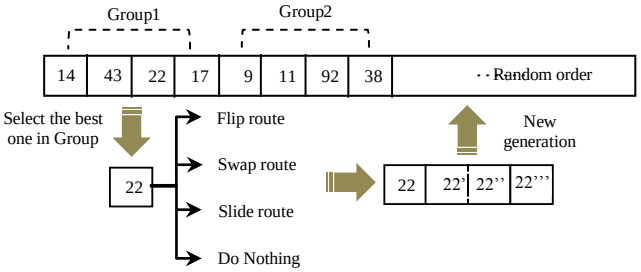

Fig. 4. The intra-group evolution operation.

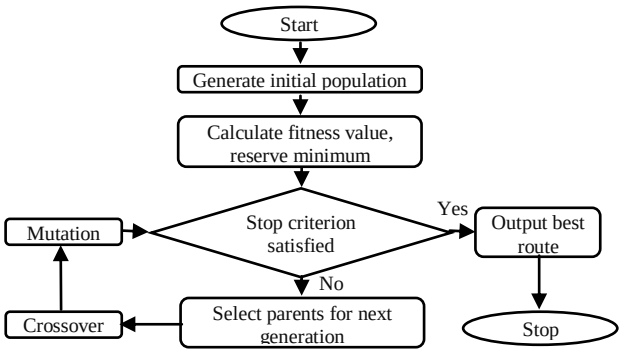

Fig. 5. The overall structure of the proposed IGA.

2.2. *Inter-group evolution operation*

The initial population was formed randomly based on the shortest loop in each cluster. Then it will be optimized by the genetic algorithm.

The population is initialized based on the rewired cluster shortest loop. First, two adjacent vertexes are chosen to be the start and the end vertex. Then a complete chromosome is formed by randomly arranging all the gene section. The next step is choosing the population size and a method for evolution. So far, the generated initial population has been built. Next, we calculate the fitness value of each chromosome and reserve the minimum. Selection is the process of choosing two parents for crossing. Crossover is to explore better solution. It is not performed on every pair of individuals and controlled by probability. Mutation is an operator which is critical to the solution of the genetic algorithm. Finally, the program stops when the criterion are satisfied.

3. Experiments

To investigate the performance of the proposed IGA based on clustering, we test the efficiency of IGA against the classical genetic algorithm (CGA) with MATLAB 2015b. we choose some benchmark test instances from TSPLIB[9]. Two comparing algorithms run 20 times for each test instance.

Table 1. Experimental results obtained by CGA and IGA under the same number of iteration.

TSP question	Population size	Iteration number	CGA		IGA	
			Best	Average	Best	Average
berlin52	50	200	9699.85	10162.13	8345.46	8541.15
	100	200	9427.18	9874.52	8254.63	8401.12
	200	200	8989.61	9474.03	8242.81	8356.11
	500	200	8799.56	9103.03	8163.27	8262.34
	1000	200	8285.21	8749.46	8084.73	8125.18
eil101	100	1000	808.61	837.09	677.02	689.88
	200	1000	767.09	791.43	673.01	684.40
	500	1000	740.58	758.59	668.25	673.64
	1000	1000	726.36	747.84	657.67	670.97
	2000	1000	714.39	722.35	656.62	665.50
bier127	100	1000	159400.49	163414.35	139944.03	145238.93
	200	1000	146106.62	153221.49	131478.22	134265.21
	500	1000	142237.61	148745.09	129671.72	130807.50
	1000	1000	140176.84	142931.92	127900.18	129944.84
	2000	1000	139835.04	142092.88	127052.11	129300.45
kro200A	100	1000	72116.92	74620.79	35230.71	36031.27
	200	1000	67067.89	69609.7	34946.30	35462.12
	500	1000	61672.51	64976.96	33543.25	34443.01
	1000	1000	59956.84	61628.26	32240.51	33296.84
	2000	1000	57244.11	59245.79	31512.77	32809.47

Table 2. Experimental results obtained by CGA and IGA under the same population size.

TSP instances	Population size	Iteration number	CGA		IGA	
			Best	Average	Best	Average
eil101	500	50	1599.41	1610.18	871.03	898.52
	500	100	1150.32	1391.23	728.04	753.72
	500	500	842.28	868.51	680.35	691.64
	500	1000	740.58	758.61	676.61	684.28
	500	2000	679.57	691.21	673.90	679.93
bier127	500	100	237123.17	269132.81	130807.25	137579.35
	500	500	164351.54	172929.27	129555.21	131478.19
	500	1000	142237.61	148745.09	125731.18	129273.77
	500	2000	127663.09	132703.63	123022.83	127551.81
	500	5000	122313.88	130210.18	120077.62	125280.82
kro200A	500	100	180921.11	177608.62	81808.22	89058.73
	500	200	137501.23	139441.09	48858.73	52588.77
	500	500	84559.72	88444.19	38819.12	38230.72
	500	1000	61672.51	64976.96	33909.47	34852.43
	500	2000	47152.68	48342.31	30636.73	31188.28
d657	1000	100	161902.88	176572.27	89979.84	91691.43
	1000	500	159170.02	162237.98	83291.11	86837.84
	1000	1000	138735.24	139983.61	79128.12	82170.33
	1000	2000	123594.08	134340.46	75864.72	76765.91
	1000	5000	109622.85	110753.05	57614.77	58399.32

Table 1 and Table 2 show the experimental results. It can be shown from the two tables that IGA performs much better than CGA. Under the same iteration, IGA can achieve a relatively better solution with a small population size while CGA need a large one. Furthermore, with the same population size, IGA can always solve the TSP effectively, especially on the large-scale test instances, the advantages of IGA are more obvious. As shown in Table 1, take berlin52 for an example, CGA achieves the result 8749.46 with 1000 population

size, and IGA just uses population size of 50 to obtain the result of 8541.15. As shown in Table 2, take eil101 for an example, when the iteration number is under 500, IGA obtains a more better results than CGA. So as other test instances.

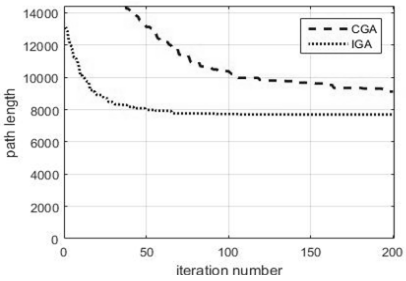

Fig. 6. Convergence curve of berlin52.

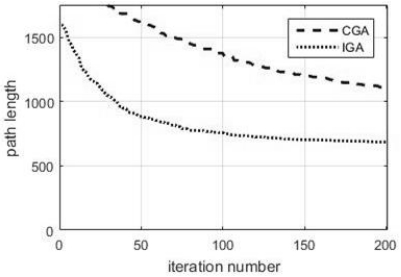

Fig. 7. Convergence curve of eil101.

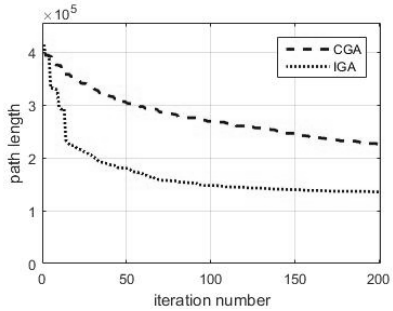

Fig. 8. Convergence curve of bier127.

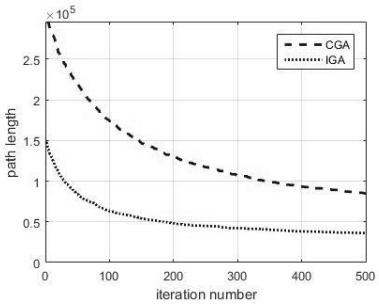

Fig. 9. Convergence curve of kro200A.

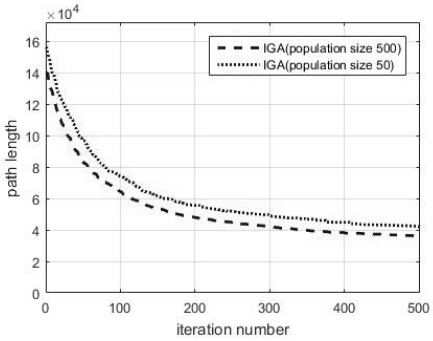

Fig. 10. Comparison between the population size of 50 and 500.

In Figure.6-Figure.9, the convergence curves of the test instances are shown. Obviously, IGA has an excellent convergence and efficiency compared with CGA.

In order to test the parameters sensitivity, we also test the performances of IGA when setting different population sizes. It can be seen from Fig.10 that, the

solutions quality and the convergence speeds of 50 and 500 bodies are almost the same. We can get that the proposed IGA is less sensitive to population size. IGA behaves well on solving TSPs, its efficient performance attributes the success to clustering.

4. Conclusion

In this study, we propose a modified genetic algorithm with *k*-means clustering for solving traveling salesman problems. Firstly, k-means clustering method is used on all the cities to reduce the complexity of the problems, through which the cities are divided into several groups. Secondly, genetic algorithm is adopted in each group for optimizing the sub-path. Thirdly, evolution operation is used on the inter-group for integral optimization.

In order to test the performance of the proposed IGA, we run a great quantity of experiments and compare with classical GA (CGA). Experimental results show that, the convergence speed of IGA is faster, and the results obtained by IGA are better than those obtained by CGA. IGA is more effective than CGA, especially for large-scale traveling salesman problems.

References

1. Samir Maity, Arindam Roy and Manoranjan Maiti, A Modified Genetic Algorithm for solving uncertain Constrained Solid Traveling Salesman Problems, *Computers & Industrial Engineering*, 83(2015).
2. Lin S, Computer solution of the traveling salesman problem. *Bell Syst Technical Report*. (MITRE Corporation,1965).
3. Davendra D. Traveling Salesman Problem, theory and applications.(In Tech, Croatia 2010).
4. Ding-Chao, Cheng-Ye, and HE-Miao, Two-Level Genetic Algorithm for Clustered Traveling Salesman Problem with Application in Large-Scale TSPs. *Tsinghua Science and Technology* 12,4 (2007).
5. Chiranjit Changdar, G.S. Mahapatra, Rajat Kumar Pal, An efficient genetic algorithm for multi-objective solid travelling salesman problem under fuzziness, *Swarm and Evolutionary Computation*, 15 (2014).
6. Yong-Deng, Yang-Liu, and Deyun-Zhou. An Improved Genetic Algorithm with Initial Population Strategy for Symmetric TSP, *Mathematical Problems in Engineering*, 3(2015).
7. Zakir Hussain Ahmed, Improved genetic algorithms for the travelling salesman problem, *Int. J. Process Management and Benchmarking*, 4, 1(2014).

8. TSPLIB,(2015),http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95.