Genetic Algorithms Based on Clustering for Traveling Salesman Problems

Lizhuang Tan, Yanyan Tan, Guoxiao Yun, Yanna Wu
School of Information Science and Engineering
Shandong Provincial Key Laboratory for Novel Distributed Computer Software Technology
Shandong Normal University
Jinan, China

***Abstract*—Genetic Algorithm (GA) is an effective method for solving Traveling Salesman Problems (TSPs), nevertheless, the Classical Genetic Algorithm (CGA) performs poor effect for large-scale traveling salesman problems. For conquering the problem, this paper presents two improved genetic algorithms based on clustering to find the best results of TSPs. The main process is clustering, intra-group evolution operation and inter-group connection. Clustering includes two methods to divide the large scale TSP into several sub-problems. One is k-means, and the other is affinity propagation (AP). Each sub-problem corresponds to a group. Then we use GA to find the shortest path length for each sub-problem. At last, we design an effective connection method to combine all those groups into one which is the result of the problem. we trial run a set of experiments on benchmark instances for testing the performance of the proposed genetic algorithm based on *k*-means clustering (KGA) and genetic algorithm based on affinity propagation clustering (APGA). Experimental results demonstrate their effective and efficient performances. Comparing results with other clustering genetic algorithms show that KGA and APGA are competitive and efficient.**

Keywords-large-scale traveling salesman problem; genetic algorithm; clustering; k-means; affinity propagation

I. INTRODUCTION

Traveling Salesman Problem (TSP) is the problem of searching for the shortest Hamiltonian tour through all the cities. TSP is a well-known NP-hard problem. It has many real world applications [1,2], such as planning scheduling, logistics distribution, computer networks and VLSI routing. Different types of TSPs have been studied by the researchers during the recent years [3-6].

The problem of TSP can be described as follows: there are N cities and distance matrix $D=\left(d_{ij}\right)_{N\times N}$ which gives distances from one city to another city. The objective of the TSP is to find the shortest route from all of the paths. A route can be seen as a cyclic permutation of cities from 1 to N if $\pi(i)$ is defined as the city visited in step i, $i=1,\ldots,N$. The cost of a route is as follows:

$$\text{minimize } f_{\pi}=\sum_{i=1}^{N-1} d_{\pi(i)\pi(i+1)}+d_{\pi(N)\pi(1)} \qquad (1)$$

If the distance satisfies $d_{ij}=d_{ji}$ for $1\le i,j\le N$, this case is the symmetric TSP.

TSP can be modeled as a weighted graph. Each vertex represents a city and each edge connects two cities. The weight of the edge represents the distance between the two connected cities. Now a TSP problem is actually a Hamiltonian cycle, and the optimal TSP path is the shortest Hamiltonian cycle.

Algorithms for solving the TSP can be summarized in two classes, exact algorithms and heuristic algorithms. The exact algorithms make sure that the final solution is optimal. Branch and cut algorithm is a typical example in this class [7,8]. The problem with these algorithms is that they are quite complex and are very demanding in computer power [9]. Since the introduction of simulated annealing [10] and tabu search [11] a breakthrough was obtained with the introduction of metaheuristics which have the possibility to find their way out of local optima. In the last twenty years, a number of nature inspired or swarm intelligence methods, like ant colony optimization [12,13], particle swarm optimization [14] and genetic algorithms [15,16] have been proposed for the solution of the TSPs.

Genetic Algorithm (GA) is an effective approach for searching optimal solution by simulating natural evolution process for problems with huge search, such as TSP. The aim of GA is to obtain an approximate solution in a large-scale problem through a couple of genetic operations like selection, crossover, and mutation. Compared with other exact search algorithms, its advantages mainly performs that the search is conducted using information of a population of tours instead of just one tour [5]. Aside from the foregoing content, the GA evaluates the 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.

Though GA is an effective method for solving TSPs, nevertheless, with the number of the traveled cities grows, classical genetic algorithm performs poor effect. In order to make the problem of TSP easier and solve the large scale TSPs efficiently, this paper presents two improved genetic algorithms with clustering, named KGA and APGA. First, KGA and APGA use clustering method to divide a large scale TSP into several sub-problems, each sub-problem corresponds to a cluster. *K*-means and affinity propagation clustering methods are respectively adopted in KGA and APGA. Then, we use GA to find the shortest Hamiltonian cycle for each cluster. All these clusters can be handled parallel. At last, we

This work is supported by the National Natural Science Foundation of China (No. 61373081, 61170145, 61401260, 61402268, 61572298), the Technology and Development Project of Shandong (No. 2013GGX10125), the Natural Science Foundation of Shandong China (No. BS2014DX006, ZR2015PF006, ZR2014FM012) and the Taishan Scholar Project of Shandong, China.

978-1-5090-4093-3/16/$31.00 ©2016 IEEE

design effective connection method to combine several clusters into one for integral optimization with the aim of shortening the whole tour.

The rest content of this paper is organized in this way: Section II presents two clustering methods including *k*-means and affinity propagation (AP). Section III describes the proposed genetic algorithm based on *k*-means clustering (KGA) and genetic algorithm based on affinity propagation clustering (APGA). Then in Section IV, experiments and comparing results are provided. Finally, we conclude this paper in Section V.

II. CLUSTERING METHODS

A. K-means Clustering

K-means is a popular unsupervised learning algorithm that is used in a wide range of applications, such as data mining, because of its simplicity [17]. The idea is to divide a set of samples into K groups (clusters), where each object has characteristics that is similar to that of another object. We choose the most distant distance within the cluster and mark it.

The algorithm needs to produce a selection of K initial center points of the cluster $C_i (i = 1, \ldots, K)$ randomly. We called it center. Firstly, calculate the distances from each object to the other cluster centers and divide the object into a cluster whose distance is the smallest. Secondly, according to the last step, recalculate every clusters center. We repeat these two steps iteratively until the centers no longer change, to achieve convergence stability. We use the Euclidean distance to compute the distance between vertices and clusters. The purpose of the clustering is to optimize the following function:

$$\text{minimize } f = \sum_{i=1}^{K} \sum_{j=1, j \in G_i}^{N} \| x_j - C_i \|^2 \tag{2}$$

where K is the number of clusters, N is the number of vertices(or cities), x_j is the coordinate of vertex j, C_i is the coordinate of the cluster i and G_i is the group of vertices belonging to cluster i.

This algorithm can obtain the shortest squared distance by moving the cluster centers around in space. The new center of a cluster is continuously updated according to all the vertices assigned to it. The formula for calculating the centers is as follows:

$$C_i = \frac{1}{|G_i|} \sum_{j=1, j \in G_i}^{N} x_j \tag{3}$$

where $|G_i|$ is the number of vertices contained in the cluster i.

Algorithm 1 presents the pseudo code for *K*-Means clustering algorithm.

```
1 Set the K cluster centers randomly;
2 repeat
3     for each vertex do
4         Calculate distance measure to each cluster;
5         Assign it to the closest cluster;
6     end
7     recompute the cluster centers positions;
8 until stop criteria are met;
```

Algorithm 1: Pseudo code of *K*-Means

B. Affinity Propagation

Clustering method based on a measure of similarity has been widely used in engineering systems and in scientific data analysis. A common approach of clustering is to divide data into several sections and to find a set of centers such that the data points and their nearest centers have the least sum of squared errors. We select centers from all the actual existence data points and name them "exemplars". *K*-means clustering method uses a set of randomly selected exemplars initially, and then iteratively optimizes those exemplars with the aim of decreasing the sum of squared errors. *K*-means clustering method is quite vulnerable to the initial selection of exemplars, so it is usually need to optimize many times with different initializations and make the effort to find a good solution. Therefore, it only works well when the amount of clusters is small and situations are good that at least one random initialization is close to good solution.

Affinity propagation (AP) is quite different from *K*-means clustering [18], it needn't to determine the number of clusters artificially before running the algorithm. It simultaneously considers all data points as potential exemplars and regards them as representative of each cluster. There are two types of message exchanged between data points in AP. It carried out alternately with two message passing steps to update the two matrices: the "responsibility" matrix and the "availability" matrix, and each takes into account a different kind of competition. Messages can be combined at any period to decide which points are exemplars. The "responsibility" $r(i,k)$ describes the degree of point i suitable for point k, that is message from i to k. The "availability" $a(i,k)$ describes the degree of data point i select the data point k as it's clustering center, send the message from i to k. Take into account the support from other points that point k should be an exemplar. $r(i,k)$ and $a(i,k)$ are calculated using the rules:

$$r(i,k) = s(i,k) - \max_{k' \neq k} \left(a(i,k') + s(i,k') \right) \tag{4}$$

$$a(i,k) = \begin{cases} \min \left\{ 0, r(k,k) + \sum_{i' \notin \{i,k\}} \max\left(0, r(i',k)\right) \right\}, i \neq k \\ \sum_{i' \neq k} \max\left(0, r(i',k)\right), i = k \end{cases} \tag{5}$$

where $s(i,k) = -\|x_i - x_k\|^2$.

Detailed description of AP can refer to [19,20].

III. GENETIC ALGORITHM BASED ON CLUSTERING

This paper presents two improved genetic algorithms with clustering, i.e., genetic algorithm based on *K*-means clustering (KGA) and genetic algorithm based on affinity propagation (APGA) for solving the large scale TSPs efficiently. First, KGA and APGA use clustering method to transform a large scale TSP into several small sub-problems, each sub-problem corresponds to a cluster. *K*-means and affinity propagation clustering methods are respectively adopted in KGA and APGA. Then, we use GA to find the shortest Hamiltonian

cycle for each cluster. All these clusters can be handled parallel. At last, we design effective connection method to combine several clusters into one for integral optimization with the objective of shortening the whole traveling route.

A. Intra-group Evolution Operation

The aim of the intra-group evolution operation is to find the shortest Hamiltonian cycle for the given vertices in each cluster. Genetic algorithm is an impactful technique based on evolution theory for problems like TSP [21]. GA is performed in each cluster aiming to obtain an approximate solution by a couple of genetic operations like selection, crossover, and mutation. Compared with other exact traditional search algorithms, its advantages mainly performs that the search is conducted using information of a population of cycles instead of just one cycle.

Ordinal encoding scheme is used in intra-group. Using this scheme, each vertex is numbered a unique integer from 1 to the number of vertices in this cluster. Chromosomes are permutations of integers, which represent the traveling paths. We define gene fragment as a permutation of the sequence numbers of vertices in a cluster. A chromosome can be considered as a permutation of all the gene fragments, and each one gene fragment represents a cluster.

The process of the genetic algorithm used in each cluster is listed as follows:

```
1 generate initial population randomly;
2 Calculate fitness value and reserve the minimum;
3 repeat
4     Select parents for next generation;
5     Perform the crossover operator;
6     Perform the mutation operator;
8 until stop criteria are met;
9 Output the best route;
```

Algorithm 2: Pseudo code of genetic algorithm used in intra-group

Genetic algorithm for solving TSP is used cluster by cluster. All those clusters can be handled parallel. The result of this step is tours $T_1, T_2, \ldots, T_k$ for clusters $G_1, G_2, \ldots, G_k$.

B. Inter-group Connection

The aim of solving TSPs is to find the shortest traveling tour. In the last step, what we have obtained is the shortest Hamiltonian cycle for the given vertices in each cluster. Then in this step, we need to consider how to connect all the clusters and obtain a whole tour.

Connect two clusters, in other words, determine which edges will be deleted from the adjacent shortest Hamiltonian cycle among each cluster, and which edges will be linked for combining two adjacent clusters into one. Suppose i and j are two closest vertices between two clusters G_i and G_j. For G_i, i-1 and i+1 are two adjacent vertices of i, and the same to G_j, j-1 and j+1 are two adjacent vertices of j. Given G_i and G_j, in order to combine the two clusters into one, we need to select two vertices $i^* \in i'$ and $j^* \in j'$ for deleting and linking edges. How to select them, we refer Eq. (6):

$$\{i^*, j^*\} = \arg\min_{i',j'} \begin{Bmatrix} d_{ij} + d_{i'j'} - d_{ii'} - d_{jj'}, \\ d_{ij'} + d_{i'j} - d_{ii'} - d_{jj'} \end{Bmatrix} \quad (6)$$

where $i' \in \{i-1, i+1\}$, $j' \in \{j-1, j+1\}$. Repeat this procedure until all clusters are joined into one whole tour. Fig.1 shows this scheme.

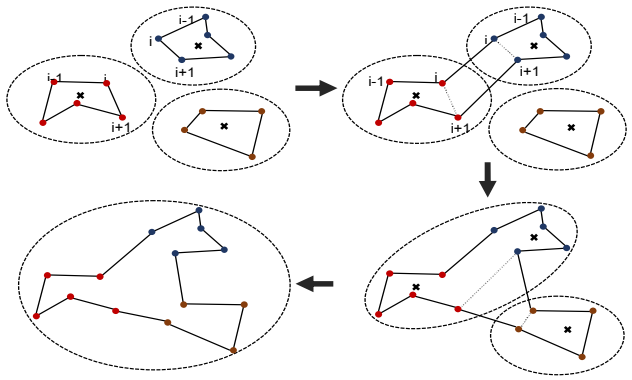

Figure 1. Process of combing clusters

Different combing sequences among clusters will result in different traveling tours, searching for the shortest is our purpose. Therefore, when the number of clusters is large, we consider designing a modified genetic algorithm for integral optimization with the aim of shortening the whole traveling route. Ordinal encoding scheme is also used in the integral optimization. However, different from chromosomes representing the traveling paths, in this process, we encode combing sequences among clusters. In other words, we need to optimize the sequence of the clusters and find the best combing sequence. Following this sequence, the first two clusters are combined into one, then the new generated cluster combines with the third cluster, and so on, step by step. At last, all those clusters are joined into one tour, and the shortest whole traveling tour is derived.

The whole process of the proposed algorithm is listed as follows:

```
1  Input an TSP;
2  K-means or AP is adopted to cluster the TSP into k sub-problems;
3  For each sub-problem i = 1 to k, do:
4      repeat
5          Select parents for next generation;
6          Perform the crossover operator;
7          Perform the mutation operator;
8      until stop criteria are met;
9      Output Hamiltonian cycle for sub-problem i;
10 End
11 Seek for the best combing sequence S with GA;
12 Combine all those Hamiltonian cycles into one tour following the optimal sequence S;
13 Output the shortest whole traveling tour.
```

Algorithm 3: Pseudo code of the proposed algorithm

IV. EXPERIMENTS

In this section, we carry out extensive experiments to evaluate the effectiveness of both KGA and APGA, which KGA represents the combination of *K*-means clustering with genetic algorithm, and APGA represents the combination of affinity propagation clustering with genetic algorithm. We have applied them on the standard test instances from TSPLIB [22] and compare their performances. Furthermore, under the same conditions, we have also compared the results with those obtained by classical genetic algorithm (CGA) and other related clustered genetic algorithms. The proposed KGA and APGA run 20 times independently for each test instance.

TABLE I presents the experimental results for each benchmark problem and statistics for 20 independent runs, including the best, mean, and standard deviation (std) of the tour length values. As described in TABLE I, the mean values obtained by KGA are respectively smaller than those obtained by APGA for att532, d657, rat783, u2319 and pcb3038, which indicates that KGA performs better than APGA on these test problems. Meanwhile, APGA performs better than KGA on the other test instances including pcb442, u2152 and rl5915. Generally speaking, KGA performs similar or a bit better than APGA in optimizing the traveling tour. However, K-means clustering is quite sensitive to the initial selection of centers, and the number of clusters needs to be set in advance. Then we prefer APGA for solving this kind of TSPs.

TABLE I. COMPARISONS OF KGA AND APGA

Problems	Optimum	KGA				APGA			
		mean	min	std	cost time(s)	mean	min	std	cost time(s)
pcb442	50778	62129.1	61461.5	279.3	5.0	**61955.6**	61946.2	257.8	7.3
rat575	6773	**7647.09**	7820.56	263.2	5.2	7814.35	7925.44	156.2	4.6
d657	48912	**56785.3**	56738.2	43.5	5.1	56801.5	56784.1	47.9	8.4
rat783	8806	**9882.9**	9822.0	44.6	6.6	9997.1	9934.7	39.6	10.7
u2152	64253	75689.3	75184.4	297.4	19.5	**75571.8**	73825.2	1807.3	49.6
u2319	234256	**243336.3**	242605.5	412.7	25.0	245704.4	249980.1	1066.5	68.9
pcb3038	137694	**159777.9**	159182.2	906.7	95.2	161086.5	160284.7	1116.5	133.6
rl5915	565530	786908.3	780619.9	7590.9	80.5	**656647.9**	619020.9	5895.5	188.6

Furthermore, we compare the proposed KGA, APGA with other related works including classical genetic algorithm (CGA) and Two-Level Genetic algorithm (TLGA) [3]. Comparisons of these four algorithms are shown in TABLE II. Experimental results obtained by CGA and TLGA are cited from [3].

Except the number of evolutionary iterations, other parameters are the same. The results in TABLE II show that the effects of both KGA and APGA are much better than CGA and TLGA within small evolutionary iterations, especially for APGA. In other words, deriving an optimal tour, under the same other parameters, KGA and APGA need less computational cost than CGA and TLGA. KGA and APGA are efficient. Further, APGA can produce a shorter tour in less iteration than the other three algorithms.

Figure 2 shows the evolution of the tour length with the number of iterations on test problems. In terms of CGA, KGA and APGA show obvious advantages over the ordinary algorithm. And the APGA can obtain a superior initial solution. These results indicate that the algorithms based on clustering converge, in terms of iterations, much faster than CGA. In other words, KGA and APGA need fewer iterations than CGA for solving an TSP well. From Figure 2 and TABLE II, we can conclude that KGA and APGA are more efficient and effective than CGA and they perform well in getting more reasonable tours in limited time for large TSPs.

Figure 3 plots the convergences of KGA with the different k. k is the number of clusters. It can be seen from the figure that with the increase of k, convergence speed of the KGA accelerates, and the quality of the results are improved. But when the number of clusters exceeds a certain value, initial solution of the intra-group evolution will be affected. In other words, the value of k depends on the scale of problem.

TABLE II. COMPARISONS WITH OTHER RELATED WORKS

Problems	Other research		Our research			
	CGA (Iterations=200)	TLGA (Iterations=200)	KGA (Iterations=20)	APGA (Iterations=20)	KGA (Iterations=200)	APGA (Iterations=200)
pcb442	$6.96*10^4$	$6.53*10^4$	$6.88*10^4$	$6.36*10^4$	$6.21*10^4$	$6.19*10^4$
rat783	$1.25*10^4$	$1.12*10^4$	$1.13*10^4$	$1.10*10^4$	$9.98*10^3$	$9.99*10^3$
dsj1000	$2.45*10^7$	$2.35*10^7$	$2.36*10^7$	$2.27*10^7$	$2.27*10^7$	$2.24*10^7$

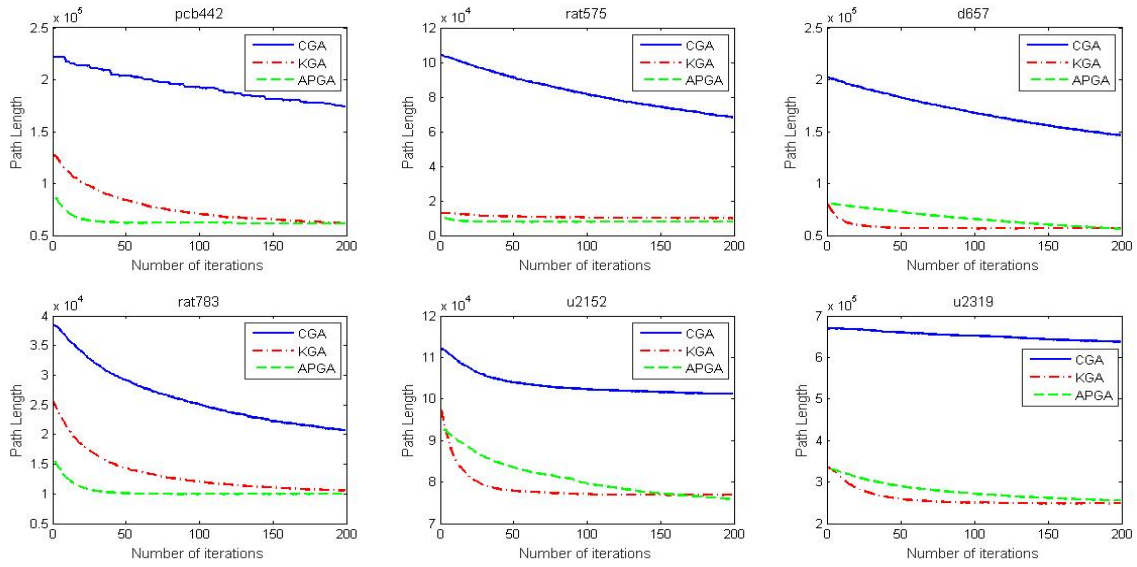

Figure 2. Comparisons of CGA, KGA and APGA

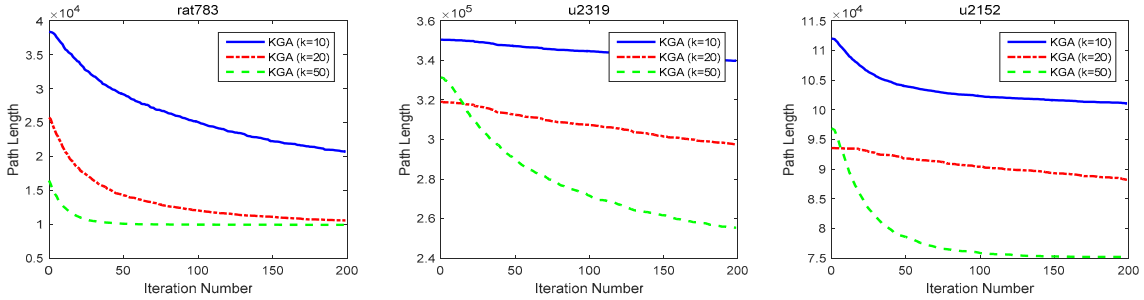

Figure 3. Comparisons of KGA with different *k*

V. CONCLUSION

In this study, we propose two improved genetic algorithms based on clustering, i.e. KGA and AGA. Their main process is clustering, intra-group evolution operation and inter-group connection. First, KGA and APGA use clustering method to separate a large scale TSP into a number of simple sub-problems, each sub-problem corresponds to a cluster. *K*-means and affinity propagation clustering methods are respectively adopted in KGA and APGA. Then, we use GA to find the shortest Hamiltonian cycle for each cluster. At last, we design an effective connection method to combine all those clusters into one for integral optimization with the aim of shortening the whole traveling route.

Experimental results demonstrate their effective and efficient performances. Comparing results with other related works show that KGA and APGA are prominent to provide reasonable results in limited iterations for TSPs.

REFERENCES

[1] F. Liu and G. Zeng, "Study of genetic algorithm with reinforcement learning to solve the TSP," Expert Systems with Applications, vol. 36, no.3, pp. 6995-7001, 2009.

[2] G. Gutin and A. P. Punnen, The Traveling Salesman Problem and Its Variations, Springer, 2002.

[3] 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, vol. 12(4), pp. 459-465, 2007.

[4] Zakir Hussain Ahmed, "Improved genetic algorithms for the travelling salesman problem," Int. J. Process Management and Benchmarking, vol. 4(1), pp. 109-124, 2014.

[5] Y. Deng, Y. Liu, and D. Zhou. "An Improved Genetic Algorithm with Initial Population Strategy for Symmetric TSP," Mathematical Problems in Engineering, vol. 3, pp. 1-6, 2015.

[6] 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, vol. 15, pp.27-37, 2014.

[7] G. Laporte, "The traveling salesman problem: an overview of exact and approximate algorithms," European Journal of Operational Research, vol. 59, pp. 231-247, 1992.

[8] D. Applegate, R. Bixby, V. Chvatal, and W. Cook, "Chained Lin-Kernighan for large traveling salesman problems," Informs Journal on Computing, vol. 15, pp. 82-92, 2003.

[9] K. Helsgaum, "An effective implementation of the Lin-Kernighan traveling salesman heuristic," European Journal of Operational Research, vol. 126, pp. 106-130, 2000.

[10] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by simulated annealing," Science, vol. 220, pp. 671-680, 1982.

[11] F. Glover, "Tabu search II," ORSA Journal on Computing, vol. 2(1), pp. 4-32, 1990.

[12] A. Liu, Z. Deng, and S. Shan, "Mean contribution ant system: an improved version of ant colony optimization for traveling salesman problem," LNCS, SEAL 2006, vol. 4247, pp.489-496, 2006.

[13] M. Dorigo, and L.M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," IEEE Transactions on Evolutionary Computation, vol. 1(1), pp. 53-66, 1997.

[14] E.F.G. Goldbarg, G.R. Souza, and M.C. Goldbarg, "Particle swarm optimization for the traveling salesman problem," EVOCOP 2006, LNCS, vol. 3906, pp.99-110, 2006.

[15] R. Baralia, J.I. Hildago, and R. Perego, "A hybrid heuristic for the traveling salesman problem," IEEE Transactions on Evolutionary Computation, vol. 5(6), pp. 1-41, 2001.

[16] H.D. Nguyen, I. Yoshihara, K. Yamamori, and M. Yasunaga, "Implementation of an effective hybrid GA for large-scale traveling salesman problems," IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetic, vol. 37, pp. 92-99, 2007.

[17] K. Wagstaff, and C. Cardie, "Constrained K-means clustering with background knowledge," in The Eighteenth International Conference on Machine Learning, 2001, pp. 577-584.

[18] Brendan J. Frey and Delbert Dueck, "Clustering by Passing Messages Between Data Points," Science, vol. 315, pp. 972-976, 2007.

[19] B. Hassanabadi, C. Shea, L. Zhang, S. Valaee, "Clustering in Vehicular Ad Hoc Networks using Affinity Propagation, Original Research Article Ad Hoc Networks," vol. 13, Part B, pp. 535-548, 2014.

[20] Optimization of Traveling Salesman Problem Using Affinity Propagatio n Clustering and Genetic Algorithm, vol.5, No.4, pp.239-245, 2015.

[21] M. Albayrak and N. Allajverdi, "Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms," Expert Systems with Applications, vol. 38(3), pp.1313-1320, 2011.

[22] TSPLIB, http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/, 2015