Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 212794, 6 pages
http://dx.doi.org/10.1155/2015/212794

Research Article

An Improved Genetic Algorithm with Initial Population Strategy for Symmetric TSP

Yong Deng,[1] Yang Liu,[2] and Deyun Zhou[1]

[1]*School of Electronics and Information, Northwestern Polytechnical University, Xian, Shaanxi 710072, China*
[2]*School of Computer and Information Science, Southwest University, Chongqing 400715, China*

Correspondence should be addressed to Yong Deng; ydeng@nwpu.edu.cn

Received 18 June 2015; Accepted 9 August 2015

Academic Editor: Chih-Cheng Hung

A new initial population strategy has been developed to improve the genetic algorithm for solving the well-known combinatorial optimization problem, traveling salesman problem. Based on the k-means algorithm, we propose a strategy to restructure the traveling route by reconnecting each cluster. The clusters, which randomly disconnect a link to connect its neighbors, have been ranked in advance according to the distance among cluster centers, so that the initial population can be composed of the random traveling routes. This process is k-means initial population strategy. To test the performance of our strategy, a series of experiments on 14 different TSP examples selected from TSPLIB have been carried out. The results show that KIP can decrease best error value of random initial population strategy and greedy initial population strategy with the ratio of approximately between 29.15% and 37.87%, average error value between 25.16% and 34.39% in the same running time.

1. Introduction

Traveling salesman problem (TSP) is a well-known NP-hard problem in many real-world applications, such as job-shop scheduling and VLSI routing [1, 2]. The aim of TSP is to find a complete, minimal-cost tour when a salesman is required to visit each of n given cities once and only once [3]. So far, TSP has often been a touchstone for new strategies and algorithms proposed to solve combinatorial optimization problem. In this paper, we consider the symmetric TSP, where the distance from city i to city j is the same as from city j to city i.

Many methods have been developed for solving TSP, including exact algorithms and approximate algorithms. The exact algorithms are carried out to find the optimal solution from all valid solutions in a number of steps. But, because of exponential complexity [3], they are always infeasible if the scale of TSP becomes large, for example, 100 cities with approximately 10^{155} different solutions. In contrast, the approximate algorithms, especially many bioinspired algorithms [4–9], can obtain accepted solutions for many NP-hard problems with (relatively) short running time. These approaches are usually very simple, like Lin-Kernigan [10], colony optimization (ACO) [11], and so on [12, 13]. All of them are efficient approaches in most of the problems, but ACO, for example, is not suitable for large-scale TSP because of its computational cost $O(mn^2)$, in which m is the ant numbers and n is the number of cities.

Genetic algorithm (GA) [14] is a global search algorithm appropriate for problems with huge search, for example, TSP, in which the initial population decides iterations, the crossover realizes the construction of the offspring, and the mutation operator maintains the diversity of the individuals. So far, there is a lot of literature to improve the effectiveness of crossover and mutation [2, 15–19]. Besides, as the first step of any GA, how to initialize an efficient population plays an important role in the process of solving a problem based on GA. Toğan and Daloğlu [20] adopted the member grouping strategy to reduce the size of the problem and the initial population strategy to reduce the number of generations. Chen et al. [21] developed a feature-based initial population method for the optimization of job-shop problem. Sharma et al. [22] proposed a domain-specific initial population strategy for compliant mechanisms. Ahmed [23] used a sequential sampling method for generating initial population.

```
// xy is a city-location matrix
t ← 1  // t is the current iterations
initialize Pop(t) with N chromosomes Pop_i(t)  // N is the pop size
while not (terminating condition) do
  for i ← 1 to N do
    f_i ← f(Pop_i(t))  // f is the fitness function
  for i ← 1 to N do
    NewPop_i(t + 1) ← randomly choose Pop_i(t) ∈ Pop(t) with p_j = f_j / Σ_{k=1}^{N} f_k
  CrossPop(t + 1) ← recombine (NewPop(t + 1)) with P_c
  //P_c is the crossover probability
  MutPop(t + 1) ← mutate (CrossPop(t + 1)) with P_m
  //P_m is the mutation probability
  Pop(t + 1) ← MutPop(t + 1)
t ← t + 1
```

Algorithm 1: A simple genetic algorithm for TSP (xy).

In this paper, a new initial population method has been developed to increase the quality of initial population. Based on this method, the application performance of GA on TSP becomes more efficient. The rest of the paper is organized as follows. In Section 2, some related background theories are presented. In Section 3, the proposed method is described. In Section 4, experimental results are evaluated. In the final section, a brief conclusion is given.

2. Preliminaries

Some basic theories are shown in this section, including TSP, GA, and k-means clustering.

2.1. Traveling Salesman Problem. TSP is one of the most widely studied combination optimization problems [3]. Mathematically, this problem can be stated as follows:

$$T_d = \sum_{i=1}^{n-1} d\left(V_i, V_{i+1}\right) + d\left(V_n, V_1\right), \tag{1}$$

where path set $\text{Path}(\pi) = \{V_1, V_2, \ldots, V_n\}$ is a permutation of cities $\{1, 2, \ldots, n\}$ and $d(V_i, V_{i+1})$ represents the distance from city V_i to city V_j. The aim of TSP is to find a path from path set $\text{Path}(\pi)$ to minimize T_d.

2.2. Genetic Algorithm. Genetic algorithm (GA) is an evolutionary algorithm based on natural election, developed by Holland [24]. It is to find approximate solutions for optimization and search problems by computer simulation [25]. The aim of GA is to achieve better results through selection, crossover, and mutation. Selection is to select the best solutions preferentially according to the fitness function from the population. The fitness function ($f(T_d)$) is defined over the genetic representation and measures the quality of the represented solution. In this paper, $1/T_d$ is assigned to $f(T_d)$. Crossover and mutation are used to generate a second generation population of solutions from those selected. Crossover can vary the population from one generation to the next by recombining "parent" solutions. Mutation alters some gene of one solution to avoid local optimal solutions. In general, the basic steps of GA for a problem are shown in Algorithm 1.

2.3. k-Means Clustering. k-means clustering is as well-known as Lloyd's algorithm [26], which aims to find k centers from n observations to minimize the mean distance from each observation to their nearest center. More details about the concepts and description of k-means clustering can be found in a lot of literature [26–28]. In this paper, we present a simple k-means clustering to the initial population.

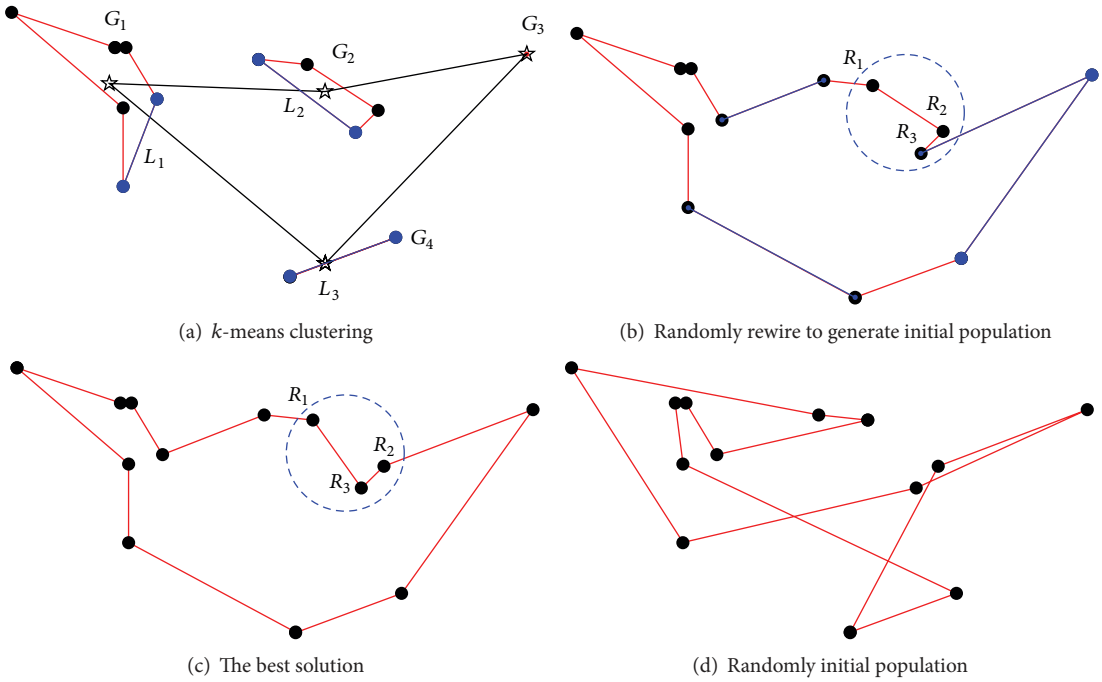
3. Proposed Method

The first step of GA is to generate an initial population in which a set of possible solutions is contained. The quality of this population plays an important role in solving a problem by GA [20, 29]. As can be seen from Figure 1, in which a TSP of 14 cities is considered, Figure 1(d) is the best solution obtained from 3000 solutions generated randomly (RIP) [30] and Figure 1(b) is the best solution from 3000 alternatives with the proposed method (KIP), respectively. Figure 1(c) is the best solution of this problem.

KIG is developed for TSP solution based on k-means clustering and GA. k-means clustering is used to divide a large-scale TSP into some small problems to obtain local optimal solutions. Then, GA is carried out to globally optimize the alternatives generated by randomly rewiring each local optimal solution. Without loss of generality, consider a TSP with N cities, in which (x_i, y_i) denotes the location of city i, $i = 1, 2, \ldots, N$. An initial population can be obtained as follows.

Step 1. N cities cluster into K groups with $K = [\sqrt{N} + 0.5]$ based on k-means clustering.

Step 2. GA is used to obtain the local optimal path of each group and a global optimal path of K groups.



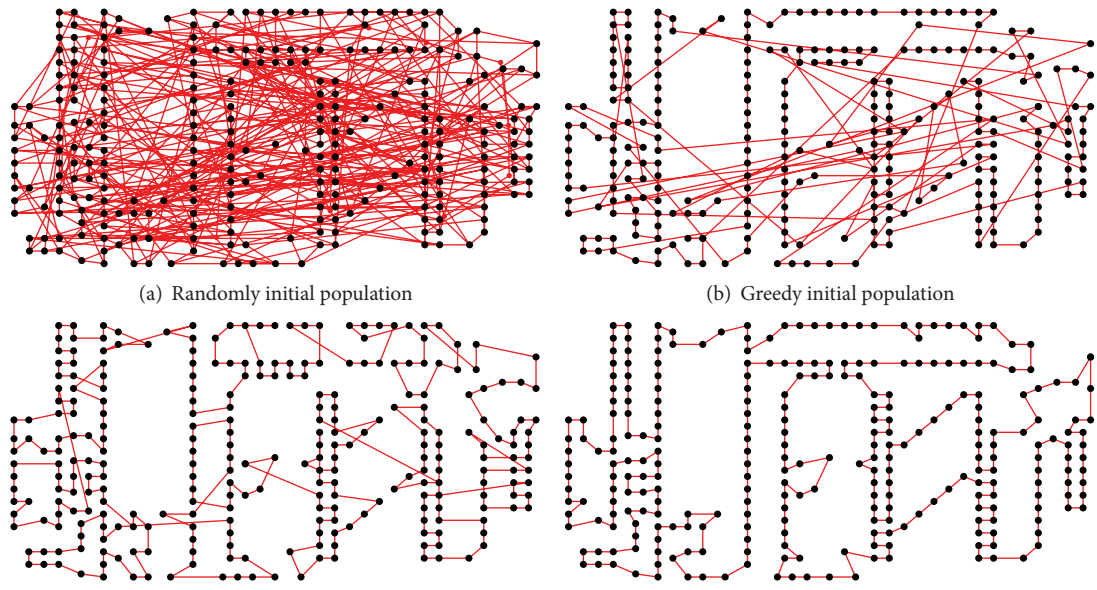
(a) k-means clustering

(b) Randomly rewire to generate initial population

(c) The best solution

(d) Randomly initial population

FIGURE 1: The processes to initialize the population with k-means clustering. Firstly, 14 nodes are clustered into 4 groups. Secondly, GA is used to obtain the local optimal path of each group and a global optimal path of 4 groups. Finally, disconnect and rewire to obtain the initial population.



(a) Randomly initial population

(b) Greedy initial population

(c) k-means clustering initial population

(d) The best solution

FIGURE 2: Three methods to initialize the population of problem a280 [35]. (a) is RIG, (b) is generated by a greedy method (GIP) [25], (c) is KIP, and (d) is the best solution of this problem.

Step 3. According to the global optimal path, one edge of each local optimal path disconnects to rewire the front and back groups.

Step 4. Repeating Step 3, an initial population can be generated.

Considering the 14 cities example as shown in Figure 1, 14 cities cluster into 4 groups G_1, G_2, G_3, G_4. A global optimal path $\{G_1, G_2, G_3, G_4, G_1\}$ (the black path) and 3 local optimal paths (the red and blue paths) can be seen from Figure 1(a). Then, select one edge from each group (L_1, L_2, L_3) and disconnect to rewire the front and back group, such as Figure 1(a)

```
// xy is a city-location matrix
t ← 1  // t is the current iterations
initialize Pop(t) with N chromosomes Pop_i(t)  // N is the pop size
while not (terminating condition) do
  for i ← 1 to N do
    f_i ← f(Pop_i(t))  // f is the fitness function
  for i ← 1 to N do
    Pop_i(t + 1) ← choose Pop_k(t) satisfy f_k > f_j, j ∈ {i, i + 1, i + 2, i + 3} and j ≠ k
    Pop_{i+1}(t + 1) ← mutate Pop_i(t + 1) by flip mutation method [31]
    Pop_{i+2}(t + 1) ← mutate Pop_i(t + 1) by swap mutation method [32]
    Pop_{i+3}(t + 1) ← mutate Pop_i(t + 1) by slide mutation method [33]
    i ← i + 4
t ← t + 1
```

Algorithm 2: An improved genetic algorithm for TSP (xy) developed by Kirk [34].

Table 1: Results of different strategies with the same iterations.

Method	RIP			GIP			KIP		
Examples	Best err. (%)	Ave. err. (%)	Ave. time (s)	Best err. (%)	Ave. err. (%)	Ave. time (s)	Best err. (%)	Ave. err. (%)	Ave. time (s)
berlin52	**0.0000**	5.1418	16.9338	**0.0000**	**1.9531**	18.2832	**0.0000**	3.0120	**16.3208**
kroA100	2.3761	4.8328	**20.3049**	0.2787	4.0082	22.5599	0.1035	2.5505	20.3318
pr144	**1.3889**	4.7425	**23.4091**	3.1847	6.0775	26.6297	2.0176	**3.5041**	26.5335
ch150	5.6974	9.5032	**23.8436**	**4.1780**	**5.9298**	27.4067	5.1038	6.7850	25.4257
kroB150	**2.9646**	6.4505	**23.8947**	6.3054	8.2309	27.4617	3.3544	**5.8072**	25.5628
pr152	**1.4383**	4.4424	**24.1544**	3.1397	4.2628	27.4174	2.5998	**3.5800**	26.4204
rat195	**6.7733**	**9.2618**	**27.2911**	7.8855	11.2154	30.5484	8.7133	11.3242	30.0717
d198	3.9011	6.8199	**27.5327**	5.1889	6.3098	32.1868	**2.8149**	**4.5911**	32.5780
kroA200	10.4079	13.3538	**27.8636**	11.1678	12.2504	32.4812	**5.3725**	**8.8791**	31.8691
ts225	16.0522	17.9204	**29.5267**	14.6896	15.499	33.6234	**10.3008**	**14.5409**	34.1776
pr226	5.2754	9.1196	**29.7816**	8.6261	12.9046	35.4601	**4.9741**	**6.1991**	36.3008
pr299	20.1269	24.5098	**35.0112**	29.1241	30.3129	42.5489	**14.6584**	**17.0753**	44.6964
lin318	30.1053	34.0994	**36.2227**	24.9692	31.5281	45.8366	**15.0669**	**17.2412**	47.2462
pcb442	54.7724	58.4570	**45.8070**	44.7984	49.0039	57.0011	**22.5887**	**24.0461**	62.8187
Average	11.5200	14.9039	**27.9698**	11.6812	14.2490	32.8175	**6.9763**	**9.2240**	32.8824

(disconnect) to Figure 1(b) (rewire). From Figure 1(b) (the best solution from the initial population) to Figure 1(c) (the best solution of this problem), one step ($\{R_1, R_2, R_3\} \rightarrow \{R_1, R_3, R_2\}$) can be achieved. Besides, a performance about three methods on example a280 selected from TSPLIB [35] is shown in Figure 2.

4. Experimental Results

We conduct the experiments on a computer with Intel Core i3-2120 3.30 GHz processor using MATLAB R2013a. To test the efficiency of the proposed method, 14 different TSP are taken from the TSPLIB [35] and 10 trials are examined for each problem. Without loss of generality, an improved GA developed by Kirk [34] was used to test the efficiency of the proposed method. The basic steps of this improved method are shown in Algorithm 2.

Firstly, the same iteration (20000) is considered to analyze the time cost and the quality of the solutions. There is no crossover but three mutations occurred 100% at each iteration in the software. As we can see from Table 1, KIP is superior to other methods: 64.29% of best error results and 78.57% of average error results. Then, the same running time is used to analyze the iterations and the quality of the solutions by three methods on each problem. 64.29% of best error results and 78.57% of average error results demonstrate KIP more efficiently as shown in Table 2. Besides, Figure 3 shows the best solution varies by time about example pcb442.

5. Conclusion

In this paper, we present a new initial population strategy (KIP) to improve GA that is used to find the optimal solution for the well-known traveling salesman problem (TSP). To test the performance of this strategy, 14 different TSP examples

Table 2: Results of different strategies with the same time.

Method		RIP			GIP			KIP	
Examples	Best err. (%)	Ave. err. (%)	Ave. time (s)	Best err. (%)	Ave. err. (%)	Ave. time (s)	Best err. (%)	Ave. err. (%)	Ave. time (s)
berlin52	2.2890	5.6054	16.3333	4.5497	5.4338	16.3333	**0.0000**	**3.0120**	16.3208
kroA100	2.4233	4.8220	20.3425	0.7447	4.4327	20.3425	**0.1035**	**2.5505**	20.3318
pr144	**1.5751**	**2.9365**	26.5358	1.8611	4.8923	26.5358	2.0176	3.5041	26.5335
ch150	**4.7941**	9.8965	25.4282	5.0946	7.7747	25.4282	5.1038	**6.7850**	25.4257
kroB150	**2.9470**	5.9548	25.5686	3.4723	6.9908	25.5686	3.3544	**5.8072**	25.5628
pr152	3.0562	4.2483	26.4266	1.7454	3.9500	26.4266	**2.5998**	**3.5800**	26.4204
rat195	**6.3251**	**8.6355**	30.0770	11.4580	12.9902	30.0770	8.7133	11.3242	30.0717
d198	4.4745	5.3608	32.5886	4.4171	4.9603	32.5886	**2.8149**	**4.5911**	32.5780
kroA200	9.3804	11.7009	31.8710	8.4947	11.349	31.871	**5.3725**	**8.8791**	31.8691
ts225	**9.7628**	**13.5387**	34.1798	11.1856	16.8642	34.1798	10.3008	14.5409	34.1776
pr226	6.6253	7.0874	36.3014	9.6295	12.8062	36.3014	**4.9741**	**6.1991**	36.3008
pr299	18.0739	21.8767	44.7099	24.5210	28.4839	44.7099	**14.6584**	**17.0753**	44.6964
lin318	25.4203	27.0580	47.2527	28.4007	29.9471	47.2527	**15.0669**	**17.2412**	47.2462
pcb442	40.6996	43.8170	62.8216	41.6218	45.9415	62.8216	**22.5887**	**24.0461**	62.8187
Average	9.8462	12.3242	32.8884	11.2283	14.0583	32.8884	**6.9763**	**9.2239**	32.8824

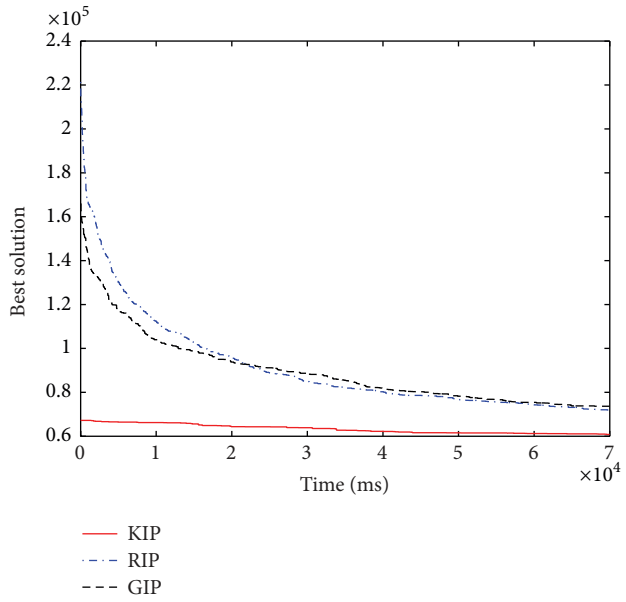


Figure 3: Three strategies, for example, pcb442.

selected from TSPLIB and two other methods (RIP and GIP) are used to experimentalize. The results show that the proposed method is efficient: 64.29% of best error results and 78.57% of average error results (Tables 1 and 2). KIP can decrease best error value of RIP and GIP with the ratio of approximately between 39.44% and 40.28%, average error value between 35.27% and 38.11% in the same iterations (Table 1). KIP can decrease best error value of RIP and GIP with the ratio of approximately between 29.15% and 37.87%, average error value between 25.16% and 34.39% in the same running time.

Conflict of Interests

No conflict of interests exists between the authors regarding the publication of this paper.

Acknowledgments

The work is partially supported by National High Technology Research and Development Program of China (863 Program) (Grant no. 2013AA013801) and the open funding project of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (Grant no. BUAA-VR-14KF-02).

References

[1] G. Gutin and A. P. Punnen, *The Traveling Salesman Problem and Its Variations*, vol. 12, Springer, 2002.

[2] 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.

[3] K. Helsgaun, "Effective implementation of the Lin-Kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

[4] E. Ruiz, M. Albareda-Sambola, E. Fernández, and M. G. C. Resende, "A biased random-key genetic algorithm for the capacitated minimum spanning tree problem," *Computers & Operations Research*, vol. 57, pp. 95–108, 2015.

[5] X. Zhang, Y. Zhang, Y. Hu, Y. Deng, and S. Mahadevan, "An adaptive amoeba algorithm for constrained shortest paths," *Expert Systems with Applications*, vol. 40, no. 18, pp. 7607–7616, 2013.

[6] H.-X. Huang, J.-C. Li, and C.-L. Xiao, "A proposed iteration optimization approach integrating backpropagation neural network with genetic algorithm," *Expert Systems with Applications*, vol. 42, no. 1, pp. 146–155, 2015.

[7] X. Zhang, S. Mahadevan, and Y. Deng, "*Physarum*-inspired applications in graph-optimization problems," *Parallel Processing Letters*, vol. 25, no. 1, Article ID 1540005, 2015.

[8] A. Adamatzky, L. Bull, and B. D. L. Costello, *Unconventional Computing 2007*, Luniver Press, 2007.

[9] X. Zhang, A. Adamatzky, F. T. Chan et al., "A biologically inspired network design model," *Scientific Reports*, vol. 5, Article ID 10794, 2015.

[10] D. S. Johnson and L. A. McGeoch, "The traveling salesman problem: a case study in local optimization," in *Local Search in Combinatorial Optimization*, vol. 1, pp. 215–310, John Wiley & Sons, 1997.

[11] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *BioSystems*, vol. 43, no. 2, pp. 73–81, 1997.

[12] N. Mladenović, R. Todosijević, and D. Urošević, "Two level General variable neighborhood search for attractive traveling salesman problem," *Computers & Operations Research B*, vol. 52, pp. 341–348, 2014.

[13] K. Karabulut and M. Fatih Tasgetiren, "A variable iterated greedy algorithm for the traveling salesman problem with time windows," *Information Sciences*, vol. 279, pp. 383–395, 2014.

[14] H. Braun, "On solving travelling salesman problems by genetic algorithms," in *Parallel Problem Solving from Nature*, pp. 129–133, Springer, 1991.

[15] N. K. Pareek and V. Patidar, "Medical image protection using genetic algorithm operations," *Soft Computing*, 2014.

[16] S. Roy, "Genetic algorithm based approach to solve travelling salesman problem with one point crossover operator," *International Journal of Computers & Technology*, vol. 10, no. 3, pp. 1393–1400, 2013.

[17] M. Thakur, S. S. Meghwani, and H. Jalota, "A modified real coded genetic algorithm for constrained optimization," *Applied Mathematics and Computation*, vol. 235, pp. 292–317, 2014.

[18] X. Zhang, Y. Deng, F. T. S. Chan, P. Xu, S. Mahadevan, and Y. Hu, "IFSJSP: a novel methodology for the job-shop scheduling problem based on intuitionistic fuzzy sets," *International Journal of Production Research*, vol. 51, no. 17, pp. 5100–5119, 2013.

[19] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "A new genetic algorithm for solving optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 27, pp. 57–69, 2014.

[20] V. Toğan and A. T. Daloğlu, "An improved genetic algorithm with initial population strategy and self-adaptive member grouping," *Computers & Structures*, vol. 86, no. 11-12, pp. 1204–1218, 2008.

[21] J. Chen, S.-Y. Zhang, Z. Gao, and L.-X. Yang, "Feature-based initial population generation for the optimization of job shop problems," *Journal of Zhejiang University SCIENCE C*, vol. 11, no. 10, pp. 767–777, 2010.

[22] D. Sharma, K. Deb, and N. N. Kishore, "Domain-specific initial population strategy for compliant mechanisms using customized genetic algorithm," *Structural and Multidisciplinary Optimization*, vol. 43, no. 4, pp. 541–554, 2011.

[23] Z. H. Ahmed, "The ordered clustered travelling salesman problem: a hybrid genetic algorithm," *The Scientific World Journal*, vol. 2014, Article ID 258207, 13 pages, 2014.

[24] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[25] M. Albayrak and N. Allahverdi, "Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms," *Expert Systems with Applications*, vol. 38, no. 3, pp. 1313–1320, 2011.

[26] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[27] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, 2002.

[28] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[29] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, "Genetic algorithms for the travelling salesman problem: a review of representations and operators," *Artificial Intelligence Review*, vol. 13, no. 2, pp. 129–170, 1999.

[30] E. Osaba, R. Carballedo, F. Diaz, and A. Perallos, "Analysis of the suitability of using blind crossover operators in genetic algorithms for solving routing problems," in *Proceedings of the 8th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI '13)*, pp. 17–22, IEEE, Timisoara, Romania, May 2013.

[31] R. Hinterding, "Gaussian mutation and self-adaption for numeric genetic algorithms," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 384–388, IEEE, December 1995.

[32] G. Ulusoy, F. Sivrikaya-Şerifoğlu, and Ü. Bilge, "A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles," *Computers & Operations Research*, vol. 24, no. 4, pp. 335–351, 1997.

[33] W. Sun, "A novel genetic admission control for real-time multiprocessor systems," in *Proceedings of the International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT '09)*, pp. 130–137, IEEE, December 2009.

[34] J. Kirk, *Traveling Saleman Problem Genetic Algorithm*, 2011, http://www.mathworks.com/matlabcentral/fileexchange/13680-traveling-salesman-problem-genetic-algorithm.

[35] TSPLIB, 2014, https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/.