A new genetic algorithm based on graph edges, for solving TSP

Conference Paper · February 2017

CITATIONS
0

READS
334

2 authors:

Nasrin Bastani
Isfahan University of Medical Sciences
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Mehdi Jabalameli
University of Isfahan
4 PUBLICATIONS **3** CITATIONS
SEE PROFILE

FET 2017 25 February 2017
1st national Conference
On Future Of Engineering And Technology
Iran - Tehran

A new genetic algorithm based on graph edges, for solving TSP

Nasrin Bastani
School of Computer Engineering, Islamic Azad University of Najafabad, Najafabad, Iran
Student Research Committee, School of Advanced Technologies in Medicine, Isfahan University of Medical Sciences, Isfahan, Iran
Nasrin.bstn@gmail.com

Mehdi Jabalameli
School of Computer Engineering, Islamic Azad University of Najafabad, Najafabad, Iran
Department of Computer Engineering, University of Isfahan, Isfahan, Iran
jabalameli@eng.ui.ac.ir

Abstract

Traveling salesman problem (TSP) is an optimization problem classified in NP-Hard problems. This problem has a wide use in practical applications like urban transportation system. There are two approaches for optimization problem. Exact methods and approximation methods. Exact methods are good choices for small-scale problems due to large time complexity related to them while approximation methods are suitable for large-scale problems. Genetic algorithm is an approximation method uses nature patterns to find the solution and its basis is making a generation by optimized characteristics inherited from previous populations. In incomplete graphs, most of the algorithms make edges with infinity weight between city pairs that there is no direct path between them. Because computers have limited memories, they use large numbers instead of infinity values and causes problem in full automated algorithms that use TSP solution in a part of their process. In this paper, we proposed a new genetic algorithm for TSP which is basically different with existed genetic algorithms. Our genetic algorithm is defined based on existed edges in reference graph; this property enables algorithm to be processed based on real answers and returning true solution in incomplete graphs and also the possibility of distinguishing if there is no tour available in the graph. This is a helpful characteristic for automated algorithms uses TSP solution in their process.

Keywords: Traveling salesman problem, TSP, Genetic algorithm, approximation approach

Introduction

The salesman problem is a NP-Hard problem(Karp, 1972) which is about a salesman who wants to have a tour between n cities and started his path from one city, the goal is by passing every city just one time, and he has to return to the start point. The problem is the order of passing the cities to have the minimum distance. The assumption is distances between all city pairs are known. A tour with the size of n is defined by $tour = (v_1, v_2, v_3, \dots, v_n, v_1)$ and v_k is the kth city the salesman enters. The tour also can be defined using edges: $tour = \{(v_1, v_2), (v_2, v_3), \dots, (v_n, v_1)\}$.

There are lots of branches of TSP problem; Tour-TSP and path-TSP. the tour-TSP forces the problem to be finished with the start city while in path-TSP, the salesman is not forced to return to the first city (Papadimitriou, 1977). The other, is symmetric and asymmetric one, showing the path from v_i to v_j by p_{ij}, if $p_{ij} = p_{ji}$, for all city pairs, the problem is symmetric, otherwise it is named asymmetric TSP (ATSP) (Bai *et al*, 2013). In attractive traveling salesman, the solution is a tour with maximum profit, with the length of less than or equal to the given distance (Mladenović, Todosijević and Urošević, 2014). In Generalized TSP (GTSP), all the cities are divided to various clusters and the objective is finding minimum tour cost visiting one and just one city from each cluster (Snyder and Daskin, 2006). Physical TSP (PTSP) is defined using a ship should meet a set of waypoints scattered surrounding a 2 dimensional maze, full of obstacles (Padberg *et al*, 1980). TSP with time windows (TSPTW) need a Hamilton tour through graph nodes but each node must be visited in a delimited time window (López-Ibánez *et al*, 2013). Euclidean TSP is the shortest route on a set of points relining in a d-dimensional Euclidean space (Uğur *et al*, 2009). In the consistent TSP (conTSP) the problem is finding minimum rout in a multi-period routing (Subramanyam and Gounaris, 2016).

The most famous type of TSP is symmetric TSP.TSP can be used in a wide range of practical problems like city transportation system, transforming distribution system, Data transmitting in computer network, design of optical fiber, wallpaper cut, crystallography, dartboard design or in medical applications like in laparoscopy surgeries, and so on (Laporte, 1992; Ali and Kamoun, 1993; Vega-Rodriguez *et al*, 2005; Falcone, Chen and Hamad, 2013).

There is lots of tries to solve the problem; one of the first possible solution is testing all the possibilities which there is (n-1)! Possible tours which takes a large time to find the solution. Except that, There are two main approaches for solving TSP: Exact methods like branch and bound based algorithms (Held and Karp, 1970; Padberg and Rinaldi, 1987), branch and cut based methods (Crowder and Padberg, 1980; Grötschel and Holland, 1991; Fischetti, Salazar González and Toth, 1997) and dynamic programming (Bellman, 1962; Ergun and Orlin, 2006); and approximation methods containing space filing carve based method (Hsieh and You, 2012), or nearest neighborhood algorithm(Flood, 1956). The other approximation methods branch is Nature based algorithms such ant colony optimization (Cheng and Mao, 2007; Chen and Chien, 2011a, 2011b; Olivas, Valdez and Castillo, 2014), bee colony

optimization (Karabulut and Tasgetiren, 2012) and genetic algorithm (Jiao and Wang, 2000; Onwubolu and Clerc, 2004).
For small size problems, we can use exact methods and due to large time complexity of exact methods, for large size ones, an approximation method is a good selection (Chatterjee, Carrera and Lynch, 1996; Majumdar and Bhunia, 2011).

Genetic algorithm is an approximation algorithm which is one of the most popular methods used in optimization problems. This algorithm is formed by inspiration of nature and is based on random search theory. In this algorithm, optimization is done by the principle that in a population, only solutions with better characteristics, can continue living and next generation inherit these characteristics(Bryant and Benjamin, 2000). The algorithm is done by concepts of chromosome and gene, chromosome is one of the individual solutions and gene is one of the elements the chromosome has consisted of. There are two main operation in genetic algorithm: fitness function and generation production including crossover and mutation. Fitness function calculates cost of current chromosome; Crossover and mutation, produces new population using former generation including chromosomes.
One of the problems can be mentioned for the existed method is defining infinity cost for city pairs which there is no rout between them, so when we want to program the algorithm we have to define a very large number instead of infinity value for computer. If there is no tour in the input graph, the algorithm returns a large number instead of distinguishing that there is no answer for the graph. This can be a problem for automated uses the algorithm.
In this paper, we proposed a new genetic algorithm for solving symmetric TSP, which in it, chromosomes are based on edges, not vertexes; and has the ability of determination if there is no tour in the given graph. Rest of the paper organized as follow. In the next section we present the proposed method, third section has an example solution produced by the algorithm, in second four, results are presented and in the last section we have conclusion and discussion about it.

Proposed Method

The proposed algorithm is a genetic algorithm based on existed edges in reference graph. In the first step, chromosome structure is constructed by a series of genes based on graph edges. Each chromosome contains n genes, n is the number of cities in the problem; each gene is an existed edge between two cities. The first generation production is in a random selection process. Every chromosome contains a series of edges that represents estimated solution.

Fitness function has two elements, one element is the typical TSP minimizing condition as in (1),

$$E_1: tour = \arg\min \sum_{i=1}^{n} |d_i - 2| \quad (1)$$

Which d_i is degree of vertex i in the current chromosome. And the second element is defined by (2),

$$E_2: tour = \arg\min \sum_{i=1}^{n} edge_i \quad tour = \{edge_1, edge_2, \dots, edge_n\} \quad (2)$$

And $edge_i$ is ith edge existed in the chromosome. The priority of first element is more than the first because we try to guarantee the correct existed tour. We define fitness function using (3),

$$F(chromosome) = \begin{cases} E_1 = |d_1 - 2| + |d_2 - 2| + \cdots + |d_n - 2| \\ E_2 = W_1 + W_2 + \cdots W_n \end{cases} \quad (3)$$

A new chromosome is constructed by crossover function with the process of copying common genes from two parents with the condition of not increase degree of chromosome vertexes more than number 2; in the next step, rest of genes to n is selected by the fitness function criterion in the child chromosome, means first selected gene has to cause minimum incretion of E_1, and second is minimizing of E_2, in child chromosome. The first element has more priority like before.
Mutation function in a chromosome is done by removing an edge between two vertexes that one or both of them have a degree more than number 2 in chromosome graph or having the maximum weight in the second priority, and putting an edge instead that is between two vertexes that one of them or both of

them have a degree less or equals to number 2 in the chromosome graph, or having the minimum weight between reminded edges in problem graph in the second priority.
To guarantee the solution after finishing the algorithm, answer chromosome is surveyed in the vertex order and if the all vertexes are not in the survey, a mutation function will be executed and algorithm will be restarted.

Example Problem Solution

In this section, we explain a TSP problem on a random graph generated automatically and solved by our proposed algorithm. The steps of each iteration is presented, crossover and mutation functions are showed and fitness value is calculated in each step.
Figure 1 shows a sample graph which is input of algorithm, the graph has seven vertexes, means the chromosome length is seven. The population size is a user defined number that is five, here. Also child replacement numbers in every generation is a user parameter input and it is one replacement in each generation in this graph. These two parameters effects on the running time. Mutation was executed in a random process of iterations for one gene.

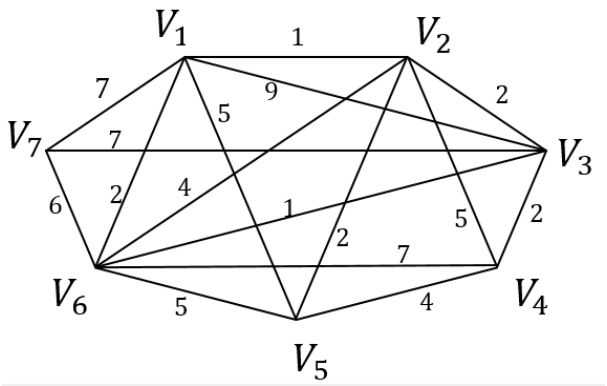

Figure 1, Sample graph for TSP solution

The first population was produced and fitness function was calculated for each chromosome (Table 1). The genes series are sorted for better understanding.
E_1 in M_4 and M_5 has minimum value among the other chromosomes, so these two chromosomes were selected for crossover function. Common genes in crossover with the condition to be transformed to the child, is listed in in Table 2.

Table 1, First population produced by the proposed genetic algorithm and their fitness function values

Chromosome	Fitness function value
$M_1 = \{(V_1,V_2),(V_1,V_3),(V_1,V_5),(V_1,V_6),(V_2,V_5),(V_2,V_6),(V_6,V_7)\}$	$F(M_1) = \begin{cases} 8 \\ 29 \end{cases}$
$M_2 = \{(V_1,V_3),(V_1,V_5),(V_1,V_7),(V_3,V_6),(V_3,V_7),(V_4,V_5),(V_5,V_6)\}$	$F(M_2) = \begin{cases} 6 \\ 38 \end{cases}$
$M_3 = \{(V_1,V_5),(V_2,V_3),(V_2,V_4),(V_2,V_5),(V_3,V_4),(V_4,V_5),(V_5,V_6)\}$	$F(M_3) = \begin{cases} 8 \\ 26 \end{cases}$
$M_4 = \{(V_1,V_2),(V_1,V_7),(V_2,V_3),(V_2,V_5),(V_2,V_6),(V_3,V_7),(V_4,V_6)\}$	$F(M_4) = \begin{cases} 4 \\ 28 \end{cases}$
$M_5 = \{(V_1,V_2),(V_1,V_7),(V_2,V_3),(V_2,V_5),(V_4,V_5),(V_5,V_6),(V_6,V_7)\}$	$F(M_5) = \begin{cases} 4 \\ 27 \end{cases}$

Table 2, Common genes between two parents that are transformed to the child chromosome in first generation

P_1	P_2	$child_1$
$(V_1, V_2), cc$	$(V_1, V_2), cc$	(V_1, V_2)
$(V_1, V_7), cc$	$(V_1, V_7), cc$	(V_1, V_7)
$(V_2, V_3), cc$	$(V_2, V_3), cc$	(V_2, V_3)
(V_2, V_5)	(V_2, V_5)	
(V_2, V_6)	(V_4, V_5)	
(V_3, V_7)	(V_5, V_6)	
(V_4, V_6)	(V_6, V_7)	

With these three genes $F_c(child_1) = \begin{cases} 8 \\ 10 \end{cases}$, which $F_c(chromosome)$ is fitness function for an uncompleted chromosome. For rest of genes, algorithm computed the increment value for fitness function in each iteration of building the completed chromosome of the child. The whole process of iterations is presented in Table 3.

Table 3, Loop of child chromosome complement

Gene	Fitness value increment				
	Iteration 1	Iteration 2	Iteration 3	Iteration 4	final
(V_2, V_5)	$F_{inc}((V_2, V_5)) = \begin{cases} 0 \\ 2 \end{cases}$	$F_{inc}((V_2, V_5)) = \begin{cases} 0 \\ 2 \end{cases}$	$F_{inc}((V_2, V_5)) = \begin{cases} 0 \\ 2 \end{cases}$	$F_{inc}((V_2, V_5)) = \begin{cases} 0 \\ 2 \end{cases}$	
(V_2, V_6)	$F_{inc}((V_2, V_6)) = \begin{cases} 0 \\ 4 \end{cases}$	$F_{inc}((V_2, V_6)) = \begin{cases} 0 \\ 4 \end{cases}$	$F_{inc}((V_2, V_6)) = \begin{cases} 0 \\ 4 \end{cases}$	$F_{inc}((V_2, V_6)) = \begin{cases} 0 \\ 4 \end{cases}$	
(V_3, V_7)	$F_{inc}((V_3, V_7)) = \begin{cases} -2 \\ 1 \end{cases}$	Selected in iteration 1			*
(V_4, V_5)	$F_{inc}((V_4, V_5)) = \begin{cases} -2 \\ 4 \end{cases}$	$F_{inc}((V_4, V_5)) = \begin{cases} -2 \\ 4 \end{cases}$	Selected in iteration 2		*
(V_4, V_6)	$F_{inc}((V_4, V_6)) = \begin{cases} -2 \\ 7 \end{cases}$	$F_{inc}((V_4, V_6)) = \begin{cases} -2 \\ 7 \end{cases}$	$F_{inc}((V_4, V_6)) = \begin{cases} -2 \\ 7 \end{cases}$	$F_{inc}((V_4, V_6)) = \begin{cases} -2 \\ 7 \end{cases}$	*
(V_5, V_6)	$F_{inc}((V_5, V_6)) = \begin{cases} -2 \\ 5 \end{cases}$	$F_{inc}((V_5, V_6)) = \begin{cases} -2 \\ 5 \end{cases}$	$F_{inc}((V_5, V_6)) = \begin{cases} -2 \\ 5 \end{cases}$	Selected in iteration 3	*
(V_6, V_7)	$F_{inc}((V_6, V_7)) = \begin{cases} -2 \\ 6 \end{cases}$	$F_{inc}((V_6, V_7)) = \begin{cases} 0 \\ 6 \end{cases}$	$F_{inc}((V_6, V_7)) = \begin{cases} 0 \\ 6 \end{cases}$	$F_{inc}((V_6, V_7)) = \begin{cases} 0 \\ 6 \end{cases}$	

The produced child in the first generation was $child_1 = \{(V_1, V_2), (V_1, V_7), (V_2, V_3), (V_3, V_7), (V_4, V_5), (V_4, V_6), (V_5, V_6)\}$ with the fitness value of $F(child_1) = \begin{cases} 0 \\ 33 \end{cases}$. In this state, the algorithm surveyed $child_1$ graph to check if it is the answer. Because of the tour was not produced, mutation function was executed by removing (V_3, V_7) and replacing (V_3, V_4) from resource graph, instead, which causes the minimum increment. The final child chromosome was $child_{f1} = \{(V_1, V_2), (V_1, V_7), (V_2, V_3), (V_3, V_4), (V_4, V_5), (V_4, V_6), (V_5, V_6)\}$, and fitness value was $F(child_{f1}) = \begin{cases} 2 \\ 28 \end{cases}$. In the next generation, M_1 chromosome was replaced by $child_{f1}$ and new population is presented in Table 4. M_1 and M_5 were selected for crossover function and the final child in the third generation was $child_{f2} = \{(V_1, V_2), (V_1, V_7), (V_2, V_3), (V_4, V_5), (V_5, V_6), (V_3, V_4), (V_6, V_7)\}$ and its fitness value was $F(child_{f2}) = \begin{cases} 0 \\ 27 \end{cases}$. After checking the solution, the algorithm returned $child_{f2}$ as the final solution.

Table 4, Second generation produced by the proposed genetic algorithm and their fitness function values

Chromosome	Fitness function value
$M_1 = \{(V_1, V_2), (V_1, V_7), (V_2, V_3), (V_3, V_4), (V_4, V_5), (V_4, V_6), (V_5, V_6)\}$	$F(M_1) = \begin{cases} 2 \\ 28 \end{cases}$
$M_2 = \{(V_1, V_3), (V_1, V_5), (V_1, V_7), (V_3, V_6), (V_3, V_7), (V_4, V_5), (V_5, V_6)\}$	$F(M_2) = \begin{cases} 6 \\ 38 \end{cases}$
$M_3 = \{(V_1, V_5), (V_2, V_3), (V_2, V_4), (V_2, V_5), (V_3, V_4), (V_4, V_5), (V_5, V_6)\}$	$F(M_3) = \begin{cases} 8 \\ 26 \end{cases}$
$M_4 = \{(V_1, V_2), (V_1, V_7), (V_2, V_3), (V_2, V_5), (V_2, V_6), (V_3, V_7), (V_4, V_6)\}$	$F(M_4) = \begin{cases} 4 \\ 28 \end{cases}$
$M_5 = \{(V_1, V_2), (V_1, V_7), (V_2, V_3), (V_2, V_5), (V_4, V_5), (V_5, V_6), (V_6, V_7)\}$	$F(M_5) = \begin{cases} 4 \\ 27 \end{cases}$

Result

We proposed a new genetic algorithm which is based on edges in the graph shape of TSP. We tested our algorithm on 40 random graphs with the range of vertex number of 10 to 60. The process of graph production follows this instruction: user gives total vertex number, N and range of edge weights to the algorithm W_{min}, W_{max}; the algorithm produces the total number of edges by (4) and weights are produced by (5).

$$EN = round(rand([0,1] \times \frac{N(N-1)}{2}) \quad (4)$$

$$W_i = round(rand([0,1] \times W_{max}) + W_{min}\ , 1 \le i \le EN \quad (5)$$

We evaluated our algorithm by error computed from (6).

$$e = \left(\frac{1}{n} \sum_{j=1}^{n} \left(\frac{tour_j^{eb} - tour_j^{opt}}{tour_j^{eb}} \right) \right) \quad (6)$$

Which n is the total number of tested graphs, $tour_j^{eb}$ is the tour weight of proposed algorithm in graph j, and $tour_j^{opt}$ is the optimum tour weight in that graph.

The average error of exanimated graphs is 5.642% which means the accuracy of 94.358%. 5 of the graphs had no available tour in, in these cases, the algorithm halted after 100 iterations and announced no solution is found.

Conclusion and Discussion

One of the approaches for TSP is approximation one which find an approximation of answer, not the optimum. The advantages of these kind of algorithms is using them in large size problems with a complexity less than exact methods(Majumdar and Bhunia, 2011). In this study, we proposed a new genetic algorithm, which is subset of approximation approaches, based on edges existed in reference graph shape, for Traveling salesman problem. We tested our algorithm on 40 random graphs and calculated average error caused by proposed algorithm. The error was less that 6% in finding the tour with the minimum cost. One of the advantages of our method is that chromosome structures are based on real edges not infinity valued edges. Also the method has the ability to identify if there is no available tour in graph, this will find a special importance when the graph is incomplete and some paths not existed between cities.

References

Ali, M. K. M. And Kamoun, F. (1993). Neural networks for shortest path computation and routing in computer networks. IEEE transactions on neural networks. IEEE, 4(6), 941–954.

Bai, J. Yang, G-K. Chen, Y-W. Hu, L-S. And Pan, C-C. (2013). A model induced max-min ant colony optimization for asymmetric traveling salesman problem . Applied Soft Computing. Elsevier, 13(3), 1365–1375.

Bellman, R. (1962). Dynamic programming treatment of the travelling salesman problem . Journal of the ACM (JACM). ACM, 9(1), 61–63.

Bryant, K. And Benjamin, A. (2000). Genetic algorithms and the traveling salesman problem . Department of Mathematics, Harvey Mudd College, 10–12.

Chatterjee, S. Carrera, C. And Lynch, L.A. (1996). Genetic algorithms and traveling salesman problems. European journal of operational research. Elsevier, 93(3), 490–510.

Chen, S-M. And Chien, C-Y. (2011a). Parallelized genetic ant colony systems for solving the traveling salesman problem. Expert Systems with Applications. Elsevier, 38(4), 3873–3883.

Chen, S-M. And Chien, C-Y. (2011b). Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. Expert Systems with Applications. Elsevier, 38(12), 14439–14450.

Cheng, C-B. And Mao, C-P. (2007). A modified ant colony system for solving the travelling salesman problem with time windows. Mathematical and Computer Modelling. Elsevier, 46(9), 1225–1235.

Crowder, H. and Padberg, M.W. (1980). Solving large-scale symmetric travelling salesman problems to optimality. Management Science. INFORMS, 26(5), 495–509.

Ergun, Ö. And Orlin, J.B. (2006). A dynamic programming methodology in very large scale neighborhood search applied to the traveling salesman problem. Discrete Optimization. Elsevier, 3(1), 78–85.

Falcone, J.L. Chen, X. And Hamad, G.G. (2013). The traveling salesman problem in surgery: economy of motion for the FLS Peg Transfer task. Surgical endoscopy. Springer, 27(5), 1636–1641.

Fischetti, M. Salazar González, J.J. And Toth, P. (1997). A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. Operations Research. INFORMS, 45(3), 378–394.

Flood, M. M. (1956). The traveling-salesman problem. Operations Research. INFORMS, 4(1), 61–75.

Grötschel, M. and Holland, O. (1991). Solution of large-scale symmetric travelling salesman problems. Mathematical Programming. Springer, 51(1), 141–202.

Held, M. and Karp, R.M. (1970). The traveling-salesman problem and minimum spanning trees. Operations Research. INFORMS, 18(6), 1138–1162.

Hsieh, Y-C. and You, P-S. (2012). A New Space-Filling Curve Based Method for the Travel-ing Salesman Problems. Appl. Math, 6(2S), 371S–377S.

Jiao, L. and Wang, L. (2000). A novel genetic algorithm based on immunity. IEEE Transactions on Systems, Man, and Cybernetics-part A: systems and humans. IEEE, 30(5), 552–561.

Karabulut, K. and Tasgetiren, M. F. (2012). A discrete artificial bee colony algorithm for the traveling salesman problem with time windows. in *Evolutionary Computation (CEC), 2012 IEEE Congress on*. IEEE, 1–7.

Karp, R.M. (1972). Reducibility among combinatorial problems. in *Complexity of computer computations*. Springer, 85–103.

Laporte, G. (1992). The traveling salesman problem: An overview of exact and approximate algorithms. European Journal of Operational Research. Elsevier, 59(2), 231–247.

López-Ibánez, M. Blum, C. Ohlmann, J.W. And Thomas, B.W. (2013). The travelling salesman problem with time windows: Adapting algorithms from travel-time to makespan optimization. Applied Soft Computing. Elsevier, 13(9), 3806–3815.

Majumdar, J. And Bhunia, A.K. (2011). Genetic algorithm for asymmetric traveling salesman problem with imprecise travel times. Journal of Computational and Applied Mathematics. Elsevier, 235(9), 3063–3078.

Mladenović, N. Todosijević, R. and Urošević, D. (2014). Two level General variable neighborhood search for Attractive traveling salesman problem. Computers & Operations Research. Elsevier, 52, 341–348.

Olivas, F. Valdez, F. and Castillo, O. (2014). A fuzzy system for parameter adaptation in ant colony optimization. in Swarm Intelligence (SIS), 2014 IEEE Symposium on. IEEE, 1–6.

Onwubolu, G.C. And Clerc, M. (2004). Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization. International Journal of Production Research. Taylor & Francis, 42(3), 473–491.

Padberg, M. and Rinaldi, G. (1987). Optimization of a 532-city symmetric traveling salesman problem by branch and cut. Operations Research Letters. Elsevier, 6(1), 1–7.

Padberg, M. Hong, S. Perez, D. Powley, E.J. Whitehouse, D. Rohlfshagen, P. Samothrakis, S. Cowling, P.I. and Lucas, S. M. (1980). Solving the physical traveling salesman problem: Tree search and macro actions. Combinatorial Optimization. IEEE, 6(1), 78–107.

Papadimitriou, C.H. (1977). The Euclidean travelling salesman problem is NP-complete. Theoretical computer science. Elsevier, 4(3), 237–244.

Snyder, L.V And Daskin, M.S. (2006). A random-key genetic algorithm for the generalized traveling salesman problem. European Journal of Operational Research. Elsevier, 174(1), 38–53.

Subramanyam, A. And Gounaris, C.E. (2016). A branch-and-cut framework for the consistent traveling salesman problem. European Journal of Operational Research. Elsevier, 248(2), 384–395.

Uğur, A. Korukoğlu, S. Çalıskan, A. Cinsdikici, M. and Alp, A. (2009). Genetic algorithm based solution for TSP on a sphere. Mathematical and computational applications. Multidisciplinary Digital Publishing Institute, 14(3), 219–228.

Vega-Rodriguez, M.A. Gutierrez-Gil, R. Avila-Roman, J.M. Sanchez-Perez, J.M. And Gomez-Pulido, J.A. (2005). Genetic algorithms using parallelism and FPGAs: the TSP as case study. in Parallel Processing, 2005. ICPP 2005 Workshops. International Conference Workshops on. IEEE, 573–579.