An Enhanced Genetic Algorithm for the Generalized Traveling Salesman Problem

Hassan Jafarzadeh
Department of Systems and Information Engineering
University of Virginia
Charlottesville, Virginia, USA
hj2bh@virginia.edu

Nazanin Moradinasab
Department of Industrial Engineering
Tarbiat Modares University
Tehran, Iran
nmn2ie@gmail.com

Milad Elyasi
Department of Industrial Engineering
Ozyegin University
Istanbul, Turkey
milad.elyasi@ozu.edu.tr

Abstract—**The generalized traveling salesman problem (GTSP) deals with finding the minimum-cost tour in a clustered set of cities. In this problem, the traveler is interested in finding the best path that goes through all clusters. As this problem is NP-hard, implementing a metaheuristic algorithm to solve the large scale problems is inevitable. The performance of these algorithms can be intensively promoted by other heuristic algorithms. In this study, a search method is developed that improves the quality of the solutions and competition time considerably in comparison with Genetic Algorithm. In the proposed algorithm, the genetic algorithms with the Nearest Neighbor Search (NNS) are combined and a heuristic mutation operator is applied. According to the experimental results on a set of standard test problems with symmetric distances, the proposed algorithm finds the best solutions in most cases with the least computational time. The proposed algorithm is highly competitive with the published until now algorithms in both solution quality and running time.**

Keywords-genetic algorithms; traveling salesman; generalized; nearest neighbor

I. INTRODUCTION

Genetic Algorithm (GA) is a search heuristic that mimics the process of natural evolution [1]. This heuristic algorithm (also sometimes called a metaheuristic) is routinely used to generate useful solutions to optimization and search problems. GAs belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. In a genetic algorithm, a population of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered [2]. The evolution usually starts from a population of randomly generated individuals and is an iterative process, with each iteration's population called a generation. In each generation, the fitness of every individual in the population is evaluated. The fitness is usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. The algorithm continues to run until one of the following criteria is met:

- A pre-defined fitness score is reached
- A specified number of iterations are completed or alternatively, a set time limit is reached
- No improvement is achieved in a specified number of iterations

The Generalized Traveling Salesman Problem (GTSP) is a generalized version of the Traveling Salesman Problem (TSP). In this problem, the cities are divided into a number of clusters and the traveling salesman has to find the minimum-cost tour which passes through exactly one city from each cluster. Consider R which is formed by a set of nodes that has been separated to m non-empty incompatible proper subsets:

$$R = \bigcup_{i=1}^{m} v_i \qquad |R| = N \tag{1}$$

Each node has been connected by one or more arcs from E, so that each arc connects two nodes from different clusters. $\forall (i,j) \in E \quad i \in v_i \;, j \in v_j \;, i \neq j$ in which R and E compose $G = (R, E)$, that represents the GTSP's net. The shortest tour that starts from one node of a cluster and passes through only one node of every other clusters and finishes at the starting point is desired.

II. LITERATURE REVIEW

The generalized form of TSP is called GTSP, and this problem's aim is optimizing the path between polygons instead of the single nodes. Authors in [3, 4] studied precision algorithms and theoretical approaches to solve this problem. An integer linear program and also, exact and heuristic separation procedure for some classes of facet-defining inequalities, which are used within a branch-and-cut algorithm for the exact solution of GTSP was proposed in [5]. In this study the best objective values of the aforementioned paper were used, in order to test the results of proposed algorithm.

Authors in [6] developed an innovative algorithm for minimizing airtime of a cutting tool. 3ρ/2 approximation algorithm has been presented in [7], where ρ is the number of cities in the largest cluster while the worst-case bound may be weak, as ρ may be quite large. Another approximation algorithm for the group Steiner tree problem has been proposed in [8]. In both [7, 8], the triangle inequality must be satisfied. Authors in [9] proposed an efficient innovative approach for solving GTSP. In their method, random-key GA is combined with a local tour improvement heuristic to improve the quality of solutions and the time required for obtaining them. A particle swarm optimization based algorithm was recently developed as well by authors in [10]. Also, an ant colony optimization (ACO) is used to solve GTSP in [11]. Authors in [12] proposed a GA with several new features, including isolated initial populations and a new reproduction mechanism based on the TSP ordered crossover operator. This new mechanism was called mrOX, for modified rotational ordered crossover. Another way to solve the GTSP is to turn it into the TSP [13]. The procedure of adaptation of a TSP neighborhood for the GTSP has been formalized in [14]. Authors in [15] proposed Memetic Algorithm to solve the GTSP where the crossover operator relies on large neighborhood search. Their main contribution was the originality of their crossover procedure. The Lin–Kernighan heuristic is known to be a very successful TSP one. A number of adaptations of Lin–Kernighan for the GTSP are presented by authors in [14]. They have discussed several approaches to adapt TSP local search for the GTSP. Authors in [16] have extended the ant colony optimization method from TSP to GTSP and based on the basic extended ACO method, they have developed an improved method by considering the group influence, besides a mutation process and a local search technique, namely 2-OPT search has been applied. Applications of the GTSP are also discussed in [17]. A local-global approach for the generalized traveling salesman problem and also an efficient algorithm for solving the problem based on genetic algorithms is proposed in [18, 19]. A new memetic algorithm to solve the symmetric and asymmetric instances of GTSP which is called GK is presented by authors in [20]. Their new memetic algorithm due to powerful local search, well-fitted genetic operators and new efficient termination condition improved the older versions and the given experimental results show its capability. Local-global approach for the GTSP is presented in [19]. A novel hybrid metaheuristic algorithm approach using genetic algorithms is described based on this approach. Computational results are reported and compared for Euclidean TSP-lib instances. Results show that the proposed hybrid algorithm leads to good solutions in a reasonable amount of time. Also, the consultant-guided search technique with a local-global approach is combined by authors in [21] in order to solve efficiently the GTSP. They use candidate lists in order to reduce the search space and they introduce efficient variants of 2-opt and 3-opt local search in order to improve the solutions. Different local search implementations e.g. 2-opt, 3-opt, k-op and swap have been compared empirically by authors in [14]. An ensemble of discrete differential evolution algorithms with parallel populations is presented in [22]. In a single populated discrete differential evolution (DDE) algorithm, the destruction and construction (DC) procedure is employed to generate the mutant population whereas the trial population is obtained through a crossover operator.

III. Mathematical Formulation

Mathematical formulation will provide us the other aspect of the problem which will be an instrumental tool for pondering. Also the IP formulation can help us towards finding other powerful algorithms in future researches. By considering these facts, researchers try to present the exhaustive and easy to understand IP formulation. Six distinct formulations of the GTSP as an integer programming are described in [23]. These IP formulations are "compact" in the sense that the number of constraints and variables is a polynomial function of the number of nodes in the problem. In this paper, a new IP formulation is described. Symbols used in the presented mathematical model are as follows:

m : number of clusters

n_i : number of nodes in i^{th} cluster $i = 1,\dots,m$

M : set of clusters $M = \{1,\dots,m\}$

N : number of nodes $N = \sum_{i=1}^{m} n_i$

$$y_{ij} = \begin{cases} 1 & \text{if } j^{th} \text{ node from } i^{th} \text{ cluster is chosen} \\ 0 & \text{o.w.} \end{cases}$$

$$x_{ijkl} = \begin{cases} 1 & \text{if there is an arc from } j^{th} \text{ node from } i^{th} \text{ cluster to } l^{th} \text{ node from } k^{th} \text{ cluster} \\ 0 & \text{o.w.} \end{cases}$$

c_{ijkl} : the cost of traveling from j^{th} node from i^{th} cluster to l^{th} node from k^{th} cluster

$$z_{ik} = \begin{cases} 1 & \text{if there is an arc from } i^{th} \text{ cluster to } k^{th} \text{ cluster} \\ 0 & \text{o.w.} \end{cases}$$

By considering the cited symbols above, the mathematical model of the problem will be composed of one objective function and some constraints. After introducing the mathematical model, the descriptions about the objective function and constraints will be given.

$$\min\ z = \sum_{i=1}^{m}\sum_{k=1}^{m}\sum_{j=1}^{n_i}\sum_{i=1}^{n_k} ci_{jkl}\, xi_{jkl}$$

$$\sum_{j=1}^{n_i} y_{ij} = 1 \quad \forall i = \{1,2,...,m\}$$

$$\sum_{k=1}^{m}\sum_{l=1}^{n_k} x_{ijkl} = y_{ij} \quad \forall k \neq i,\ \forall i = \{1,2,...,m\}, j = \{1,2,...,n_i\}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n_i} x_{ijkl} = y_{kl} \quad \forall k \neq i,\ \forall k = \{1,2,...,m\}, l = \{1,2,...,n_k\}$$

$$\sum_{j=1}^{n_i}\sum_{l=1}^{n_k} x_{ijkl} = z_{ik} \quad \forall k \neq i,\ \forall i,k = \{1,2,...,m\}$$

$$\sum_{i \in s}\sum_{k \notin s} z_{ik} \geq 1 \quad \forall s \subset M,\ s \neq \varnothing$$

$$y_{ij} \in \{0,1\}$$

$$x_{ijkl} \in \{0,1\}$$

$$z_{ik} \in \{0,1\}$$

Objective function shows the total cost of traveling that should be minimized. The first constraint emphasizes that exactly one node from each cluster should be chosen. The second and third constraints state that the maximum number of output and input arcs is one. The fourth and the fifth constraints ensure that there is not any sub-tour in travelled rout. And finally, last three definitions show that the used variables are Boolean.

IV. Genetic Algorithm Approach for GTSP

In this section, the constructive sections of genetic algorithm and the quantity of the parameters for solving the GTSP are presented. The most suitable amount for these parameters has been found by trial and error.

A. *Chromosome Representation*

Figure 1 illustrates the bipartite chromosome that has been used. Assume that m, is the number of clusters and n_1, n_2, ….., n_m represent the number of nodes in each cluster respectively. Let m=8, n_1=3, n_2=8, n_3=3, n_4=5, n_5=6, n_6=4, n_7=6, n_8=7. A possible solution is shown in Figure 1. The first row contains integers 1 to 8 that represents the sequence of clusters that have been visited by the traveling salesman. Numbers in the second row represent the city chosen from each cluster.

5	7	3	2	4	8	1	6
4	5	3	6	1	7	3	3

Fig. 1. Chromosome Representation

So this chromosome suggests that the traveling salesman should start his trip by visiting the fourth city from the fifth cluster and after finishing his job should go to the fifth city from the seventh cluster. By using this type of chromosome, there will be m! ways to arrange numbers on the first row and there are also n_i possible arrangements for the ith cluster. So, for a problem with m clusters and $n_i, \forall i = \{1,2,...,m\}$ nodes in each cluster, the bipartite chromosome has $m!\prod_{i=1}^{m} n_i!$ possible forms which are the size of the solution space.

After computing the fitness of each chromosome, the algorithm directly passes chromosome of highest fitness value to the next generation. Therefore, the best solution is never lost. The fitness function plotted by the algorithm is always ascending type.

B. *Reproduction*

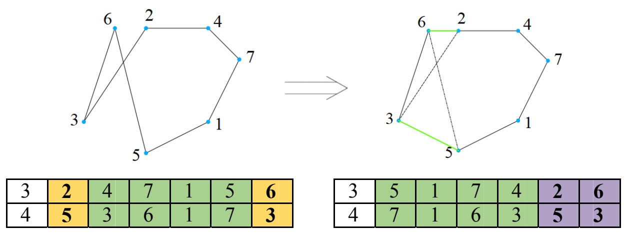
When a new population is created, the algorithm replaces 10% of the worst solutions with a random population. This way, the diversity would be preserved and a larger area of the solution space would be covered.

C. *Crossover*

In this algorithm, a simplified Greedy Selection Crossover (GSX) operator is used. This operator is adapted for the bipartite. The major merits of this operator would be revealed in problems of a larger size. In this kind of cross over, a similar city is selected arbitrarily in both chromosomes. This city is put in the first gen of offspring. In these parents, we have two cities right next to the selected first gen. Their distance from the first gen is considered and the nearest one is selected as the second gen. We will do the described step for the third city etc. The function of this operator for TSP with per mutative chromosome coding is described thoroughly by [24].

D. *Mutation*

An improving mutation operator is utilized in this algorithm. We know that an optimal path in this problem does not have any crossing segments, and mutation operator applies this theorem to promote the quality of the existing solutions. A random chromosome is picked, and each consecutive pair of its genes that makes a connecting line between two centers is analyzed for possible crossing with other pairs. If the algorithm finds such couple of pairs it exchanges those genes.



3	**2**	4	7	1	5	**6**
4	**5**	3	6	1	7	**3**

3	5	1	7	4	**2**	**6**
4	7	1	6	3	**5**	**3**

Fig. 2. Mutation

E. *Nearest Neighbor Search*

Nearest Neighbor Search (NNS) is an optimization problem for finding the closest points in metric spaces. The problem is:

Pseudocode: Nearest Neighbor Search algorithm

```
1. required c% a random chromosome
2. $\begin{bmatrix}\alpha\\ \beta\end{bmatrix} \leftarrow \lceil 1+(m-1).[rand]_{2\times 1}\rceil - 1$
3. Γ←[c(min(α,β):max(α,β))]
4. L←zeros(β-α+1) % size of matrix zeros is (β-α+1)
5. for i=1:β-α
6.     μ←∞
7.     for j=i+1:β-α+1
8.         $\delta \leftarrow \sqrt{(x_{\Gamma(i)} - x_{\Gamma(j)})^2 - (y_{\Gamma(i)} - y_{\Gamma(j)})^2}$
9.         if (δ<μ)
10              μ←δ
11.             η←ξ
12.        endif
```

```
13.  endfor
14.  Λ(i)←Γ(η)
15. endfor
16. return Λ
```

given a set of points in a metric space M and a query point $q \in M$, find the closest point in S to q. In this paper, NNS algorithm is used for promoting efficiency of GA for solving the GTSP. In the given pseudocode, c shows a sample chromosome from GA that is related randomly from the population. The algorithm generates two integer and random numbers (i.e. line 2) to pick a string of genes between them and then apply NNS on the extracted string. This string is showed by Γ in the third line. The improved string will be returned by Λ at the end of algorithm which has been pre-allocated in line 4. In the provided loop, algorithm finds the closest point to the point that already exists in the Λ. Then, it searches within the remaining points that have not been assigned yet, and it repeats these steps till reordering all the selected points as string Γ. The algorithm stops when (i) the best solution that has been obtained by algorithm, appears repeatedly in a specific number of iterations or (ii) when a specific number of iterations is performed.

V. Computational results

The number of iterations to meet the first termination criterion was set to 250 and the second criterion will differ from problem to problem, the population size iteration was set to 60, the crossover rate was set to 85% and the mutation rate was set to 5%. In every case, the best solution was obtained in less than 0.4 seconds. For investigating the efficiency of the proposed algorithm, standard test problems that are generated in [20] were solved and the results of them were collected in Tables I and II. Table I, presents the computational results in both terms of quality of solutions and CPU time. In Table II, the results of comparison between GA, Nearest Neighbor Search, GA+NNS and GK have been illustrated. GK is a state-of-the-art algorithm for solving the GTSP which is presented by [20]. "Problem" shows the standard test problem. "Best" presents the best known objective function value for each problem. "Best obtained value" is the objective value that proposed algorithm reached. "Opt" represents the number of trials that algorithm obtained the best known objective function value in 10 time runs. "Average" illustrates the mean of obtained tour length in 10 time runs. "Err (%)" represents the relative error, where the relative error is calculated as: $Err = \frac{Average - Opt}{Opt} \times 100$.

TABLE I. COMPUTATIONAL RESULTS: QUALITY OF SOLUTIONS AND TIME

Problem	Best	Best obtained value	Opt	Average	Err (%)	CPU time
10att48	5394	5394	10	5394	0.00	0.00
11eil51	174	174	10	174	0.00	0.01
12brazil53	15332	15332	10	15332	0.00	0.01
14st70	316	316	10	316	0.00	0.01
16eil76	209	209	10	209	0.00	0.01
16pr76	64925	64925	10	64925	0.00	0.01
20kroA100	9711	9711	10	9711	0.00	0.02
20rat99	497	497	10	497	0.00	0.03
20rd100	3650	3650	10	3650	0.00	0.03
21eil101	249	249	10	249	0.00	0.02
21lin105	8213	8213	10	8213	0.00	0.03
22pr107	27898	27898	10	27898	0.00	0.04
24gr120	2769	2769	10	2769	0.00	0.06
25pr124	36605	36605	10	36605	0.00	0.07
26bier127	74118	74118	10	74118	0.00	0.09
28pr136	42570	42570	10	42570	0.00	0.09
29pr144	45886	45886	10	45886	0.00	0.11
30kroA150	11018	11018	9	11018	0.01	0.08
31pr152	51576	51576	9	51576	0.01	0.16
32u159	22664	22664	9	22671	0.02	0.19
39rat195	854	854	10	854	0.00	0.24
40d198	10557	10557	10	10557	0.00	0.24
40kroA200	13406	13406	9	13406	0.01	0.36
40kroB200	13111	13111	10	13111	0.00	0.30
45ts225	68340	68340	8	68340	0.03	0.32
46pr226	64007	64007	8	64012	0.04	0.51
53gil262	1013	1013	9	1013	0.02	0.53
53pr264	29549	29549	8	29549	0.04	0.49
60pr299	22615	22615	7	22615	0.06	0.43
64lin318	20765	20765	5	20774	0.09	0.88
80rd400	6361	6361	8	6361	0.05	1.00
84fl417	9651	9651	9	9651	0.02	1.22
88pr439	60099	60099	6	60107	0.07	1.04
89pcb442	21657	21657	3	21663	0.09	1.41

TABLE II. COMPARING THE RESULT BETWEEN THE PROPOSED ALGORITHM AND GA AND NN

Problem	**Best**	**GA**		**NNS**		**GK**		**GA+NNS**	
		Err (%)	**CPU time**	**Err (%)**	**CPU time**	**Err (%)**	**CPU time**	**Err (%)**	**CPU time**
10att48	5394	0.00	0					0.00	0.00
11eil51	174	0.00	0.1	0.00	0.4			0.00	0.01
12brazil53	15332	0.00	0.3					0.00	0.01
14st70	316	0.00	0.2	0.00	0.8			0.00	0.01
16eil76	209	0.00	0.2	0.00	1.1			0.00	0.01
16pr76	64925	0.00	0.2	0.00	1.9			0.00	0.01
20kroA100	9711	0.00	0.7	0.00	7.3			0.00	0.02
20rat99	497	0.00	0.8	0.00	2.8			0.00	0.03
20rd100	3650	0.00	0.3	0.80	8.3			0.00	0.03
21eil101	249	0.00	0.2	0.40	3			0.00	0.02
21lin105	8213	0.00	0.3	0.00	3.7			0.00	0.03
22pr107	27898	0.00	0.4	0.00	5.2			0.00	0.04
24gr120	2769	0.00	0.5					0.00	0.06
25pr124	36605	0.00	0.6	0.00	12			0.00	0.07
26bier127	74118	0.00	0.5	9.68	7.8			0.00	0.09
28pr136	42570	0.00	0.5	5.54	9.6			0.00	0.09
29pr144	45886	0.00	0.3	0.00	11.8			0.00	0.11
30kroA150	11018	0.00	1.3	0.00	22.9			0.01	0.08
31pr152	51576	0.00	1.5	1.80	10.3			0.01	0.16
32u159	22664	0.00	0.6	2.79	26.5			0.02	0.19
39rat195	854	0.00	0.7	1.29	86			0.00	0.24
40d198	10557	0.00	1.2	0.60	118.8	0.00	0.14	0.00	0.24
40kroA200	13406	0.00	2.7	5.25	53	0.00	0.14	0.01	0.36
40kroB200	13111	0.00	1.4	0.00	135.2	0.00	0.16	0.00	0.30
45ts225	68340	0.00	2.4	0.00	117.8	0.00	0.24	0.03	0.32
46pr226	64007	0.00	1	2.17	67.6	0.00	0.1	0.04	0.51
53gil262	1013	0.79	1.9	1.88	122.7	0.00	0.31	0.02	0.53
53pr264	29549	0.00	1.3	5.73	147.2	0.00	0.24	0.04	0.49
60pr299	22615	0.02	6.1	2.01	281.8	0.00	0.42	0.06	0.43
64lin318	20765	0.00	3.5	4.92	317	0.00	0.45	0.09	0.88
80rd400	6361	1.37	3.5	3.98	1137.1	0.00	1.07	0.05	1.00
84fl417	9651	0.07	2.4	1.07	1341	0.00	0.73	0.02	1.22
88pr439	60099	0.23	9.1	4.02	1238.9	0.00	1.48	0.07	1.04
89pcb442	21657	1.31	10.1	0.22	838.4	0.00	1.72	0.09	1.41

Finally, "CPU time" shows the elapsed time for obtaining the best solution. Which are the times until the termination criteria are satisfied. By considering the provided results for GA and NNS algorithm in Table II, it is inferable that the combination of these two algorithms has better performance in comparison with GA and NNS alone. Based on the results of Table I, the stability of presented algorithm is visible. In the "Err (%)" column the percentage of error for most instances is zero, and in the other ones this amount is nominal. Also, the "Opt" column clarifies that the algorithm reaches the best solution in most cases.

VI. CONCLUSION

In this paper, the formulation of the GTSP has been expanded, and a GA was combined with NNS for solving it. The proposed algorithm was tested on standard test problems and the results were collected in two tables. In most runs, the best solution was obtained at a minimum time. The proposed algorithm is proven to be efficient for solving GTSP, in terms of both the quality of obtained solutions and the least running time. To evaluate the performance of the presented algorithm, its results are compared with a state-of-the-art algorithm (GK) and the quality of results is satisfactory, and also its running time is better.

REFERENCES

[1] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, 1998

[2] D. Whitley, "A genetic algorithm tutorial", Statistics and Computing, Vol. 4, No. 2, pp. 65–85, 1994

[3] G. Laporte, Y.Nobert, "Generalized traveling salesman problem through n-sets of nodes–An integer programming approach", Information Systems and Operational Research, Vol. 21, No. 1, pp. 61-75, 1983

[4] G. Laporte, H. Mercure, Y. Nobert, "Finding the shortest Hamiltonian circuit through n clusters: a lagrangean relaxation approach", Gongressus Numerantium, Vol. 48, pp. 277-290, 1985

[5] M. Fischetti, J. J. S. Gonzalez, P. Toth, "A Branch-and-cut Algorithm for the symmetric Generalized Traveling Salesman Problem", Operations Research, Vol. 45, No. 3, pp. 378-394, 1997

[6] K. Castelino, R. D'Souza, P. K. Wright, "Tool-path optimization for minimizing airtime during machining", Journal of Manufacturing Systems, Vol. 22, No. 3, 173-180, 2002

[7] P. Slavik, "On the approximation of the generalized traveling salesman problem", Rapport technique, Department of Computer Science, 1997

[8] N. Garg, G. Konjevod, R. Ravi, "A poly logarithmic approximation algorithm for the group Steiner tree problem", Journal of Algorithms, Vol. 37, No. 1, pp. 66–84, 2000

[9] L. Snyder, M. S. Daskin, "A random-key genetic algorithm for the generalized traveling salesman problem", European journal of operational research, Vol. 174, pp. 38-54, 2006

[10] X. H. Shi, Y. C. Liang, H. P. Lee, C. Lu, Q. X. Wang, "Particle swarm optimization-based algorithms for TSP and generalized TSP", Information Processing Letters; Vol. 103, No. 5, pp. 169–76, 2007

[11] K. Jun-man, Z. Yi, "Application of an Improved Ant Colony Optimization on Generalized Traveling Salesman Problem", Energy Procedia, Vol. 17, Part A, pp. 319-325, 2012

[12] J. Silberholz, B. Golden, "The generalized traveling salesman problem: a new genetic algorithm approach", Extending the Horizons: Advances in Computing, Optimization and Decision Technologies, pp. 37:165–81, 2007

[13] V. Dimitrijevic, Z. Saric, "An efficient transformation of the generalized traveling salesman problem into the traveling salesman problem on digraphs", Information Sciences, Vol. 102, No. 1-4, pp.105-110, 1997

[14] D. Karapetyan, G. Gutin, "Efficient local search algorithms for known and new neighborhoods for the generalized traveling salesman problem", European Journal of Operational Research, Vol. 219, No. 2, pp. 234-251, 2012

[15] B. Bontoux, C. Artigues, Dominique Feillet, "A Memetic Algorithm with a large neighborhood crossover operator for the Generalized Traveling Salesman Problem", Computers& Operations Research, Vol. 37, No. 11, pp. 1844-1852, 2010

[16] J. H. Yang, X. H. Shi, M. Marchese, "An ant colony optimization method forgeneralized TSP problem", Progress in Natural Science, Vol. 18, No. 11, pp. 1417–1422, 2008

[17] G. Laporte, A. Asef-Vaziri, C. Sriskandarajah, "Some applications of the generalized traveling salesman problems", Journal of the Operational Research Society, Vol. 47, No. 12, pp. 1461-1467, 1996

[18] O. Matei, P.C. Pop, "An efficient genetic algorithm for solving the generalized traveling salesman problem", 6th IEEE International Conference on Intelligent Computer Communication and Processing, pp. 87-92, 2010

[19] P. C. Pop, O. Matei, C. Sabo, "A new Approach for Solving the Generalized Traveling Salesman Problem", HM 2010, Lecture Notes in Computer Science, Vol. 6373, pp. 62-72, 2010

[20] G. Gutin, D. Karapetyan, "A Memetic Algorithm for the Generalized Traveling Salesman Problem", Natural Computing, Vol. 9, No. 1, pp. 47-60, 2010

[21] P. C. Pop, S.Lordache, "A Hybrid Heuristic Approach for Solving the Generalized Traveling Salesman Problem", GECCO, Association for Computing Machinery, pp.481-488, 2011

[22] M. Fatih Tasgetiren, P. N. Suganthan, Q. K. Pan, "An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem", Applied Mathematics and Computation, Vol. 215, No.9, pp. 3356–3368, 2010

[23] P. C. Pop, "New Integer Programming Formulations of the Generalized Traveling Salesman Problem", American Journal of Applied Sciences, Vol. 4, No. 11, pp. 932-937, 2007

[24] L. Qu, R, Sun, A" synergetic approach to genetic algorithm for solving traveling salesman problem", Information Sciences, Vol. 117, No. 3-4, pp. 267-283, 1999