A Modified TSP: Algorithms and Experimentation System

Mateusz Borowski, Rafal Machon, Iwona Pozniak-Koszalka, Leszek Koszalka, Andrzej Kasprzak
Dept. of Systems and Computer Networks
Wroclaw University of Technology
Wroclaw, Poland
{mateusz.borowski, rafal.machon}@gmail.com;
{leszek.koszalka,andrzej.kasprzak}@pwr.edu.pl

Abstract: *In this paper, the modified travelling salesman problem, in which the delivery time and the fuel consumption are taken into consideration, is studied. The objective is to create the metaheuristic algorithm which can find the best route in the defined sense. Two algorithms, which are based on Ant Colony Optimization approach and Simulated Annealing approach, are proposed. The properties of these algorithms are evaluated on the basis of the simulations made using the designed, created and implemented experimentation system. Moreover, the paper contains the results of the comparison between the proposed algorithms and the Nearest Neighbor and Random Search algorithms used as a reference. The comprehensive studies show that the proposed algorithms based on Ant Colony Optimization can ensure the greatest profit among all examined.*

Keywords: Traveling Salesman Problem, Metaheuristic Algorithm, Fuel consumption, Experimentation System

Recieved: 3 February 2015, Revised 2 March 2015, Accepted 9 March 2015

1. Introduction

Traveling salesman problem (TSP) is a very important issue nowadays and has been studied over years. There are many modifications that reflect actual problems in transport systems [4]. One of the important areas of applications of TSP is supporting logistics in courier service companies [3]. The classical task consists in finding an optimal route considering either time or distance [6]. However, there is also another significant issue, namely fuel consumption [13]. It could significantly decrease cost of company's maintenance and it is compliant with modern ecology approach. Therefore, in the last years can be observed growing interest with so-called green TSP [10] or environmental TSP [11]. Usually, the fuel consumption function is assumed to be non-linear and known in advance for specific vehicle used by couriers. For solving the modified TSP different approaches, e.g., hybrid approach [15], genetic ideas [14], or metropolis algorithm [1] can be used.

In this paper two metaheuristic algorithms for solving the formulated problem are proposed. The algorithms are based on ant colony optimization (ACO) and simulated annealing (SA) ideas. The properties of these implemented algorithms are evaluated. There are no benchmark libraries for the formulated problem available. Thus, as the reference the results given by two simple algorithms, namely nearest neighbor (NN) algorithm and random search (RS) algorithm have been taken. The investigations have been made using the created and implemented experimentation system. The user of the system can check importance of

delivery time, distance covered and fuel consumption.

The rest of the paper is organized as follows. In Section 2 the considered problem is defined in the mathematical manner. The algorithms for solving the problem are presented in Section 3 in detail. The description of the designed experimentation system is contained in Section 4. The results of the investigations made with this system are presented and discussed in Section 5. The conclusions appear in Section 6.

2. Problem Statement

The problem is some kind of a modification of TSP. The further explanation is described using a graph presented in Figure 1.

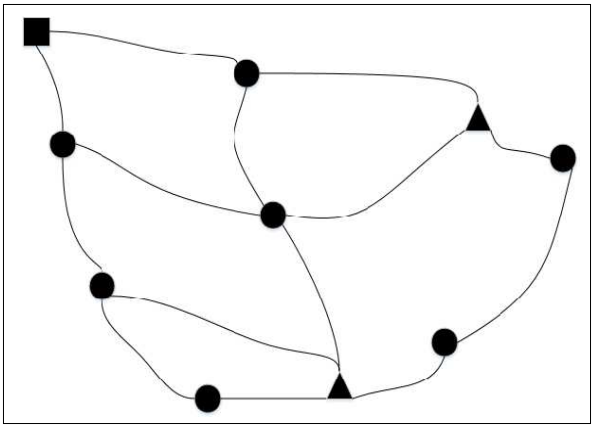

Figure 1. Exemplary graph

All nodes are denoted by circles. There is one starting node in the graph, (marked as a square). The user has to choose nodes, which are required to be visited (marked as triangles). It is assumed that: (i) each node can be visited many times, (ii) each connection between two nodes is characterized by some speed limit, and (iii) each connection between two nodes is also described by a distance that is needed to travel such a way. By the given speed and distance on certain way the traveling time could be easily computed. The objective is to visit all triangular nodes and return to the starting one (a square). It is obvious, that it can be obtained many various routes which contain all the required nodes. The aim is to choose the route that will ensure the total traveling time as short as possible, and the total fuel consumption will be as small as possible.

The considered problem can be formulated in the following way:

Given:

The graph with specified starting node (vertex) and N nodes (vertices) required for visiting.

- T – Matrix of times.
- D – Matrix of distances.
- V – Matrix of speeds.
- a_1, a_2 – Weights.
- Fuel – the nonlinear fuel consumption function, e.g., such as an example presented in Figure 2 - this function was obtained by Lagrange interpolation of earlier chosen points.

To find:
Permutation $\prod = \{\pi_1, \pi_2, \pi_n\}$, i.e., a sequence of n nodes (ID numbers corresponded to the nodes) which defines a route.

Such that:
The cost function $F(\prod)$ defined by (1) is minimal.

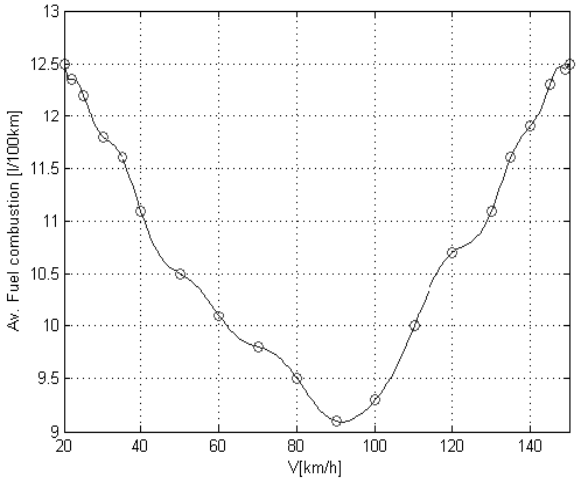

Figure 2. Fuel consumption function

$$F(\Pi) = a_1 \Sigma [T(\pi_i, \pi_{i+1})] + a_2 \Sigma \{[D(\pi_i, \pi_{i+1}) / 100] \text{ Fuel } [V(\pi_i, \pi_{i+1})]\} \quad (1)$$

In both terms the summing (denoted by Σ) is being made from $i = 1$ to $i = n - 1$. The first term of (1) represents the total traveling time of some particular route. The second term represents the average fuel consumption rate on 100 km on this route. The coefficients a_1 and a_2 can be adjusted how these costs are important for a user.

Subject to the constraint:
All *N* nodes required visiting have to be included into *n* nodes which create a permutation.

3. Algorithms

In this section the algorithms for solving the problem stated in the previous section are described, including two simple algorithms treated as reference, and two metaheuristic algorithms proposed by the authors.

3.1 Reference Algorithms

3.1.1 Random Search (RS)
This algorithm is based on the approach suggested in [12]. It works in the following way:

- **Step 1:** To choose (at random) a node from the list of the unvisited nodes .
- **Step 2:** To add the chosen node *p* to the constructed permutation Π.
- **Step 3:** To mark *p* on the list of the unvisited nodes as visited.
- **Step 4:** To go to Step 1 if the list of unvisited nodes is not empty; otherwise to stop.

3.1.2 Nearest Neighbor (NN)
This algorithm is one of the oldest approaches used to determine a solution to the travelling salesman problem [5]. The general idea is as follows: the salesman starts at a city chosen at random and repeatedly visits the nearest city until all have been visited. It quickly yields a short route, but usually not the optimal one. The algorithm can be described in the following way:

- **Step 1:** To stand on an arbitrary node as current node.
- **Step 2:** To add this node to a permutation Π.
- **Step 3:** To find out the shortest way (edge) connecting the current node and an unvisited node *p*.

• **Step 4:** To add p to Π.

• **Step 5:** To mark p on the list of the unvisited nodes as visited.

• **Step 6:** To go to Step 2 if the list of unvisited nodes is not empty; otherwise to stop.

The nearest neighbor algorithm is easy to implement and executes quickly, but it can sometimes miss shorter routes which are easily noticed with human insight, due to its "greedy" nature.

3.2 ACOBA Algorithm

Ant Colony Optimization approach was firstly introduced in [2]. An inspiration of this algorithm was behavior of ants looking for food - they are moving randomly, when they find food, they return to the colony, leaving pheromone on paths. When other ant finds pheromone, it starts follow pheromone. After some time pheromone is evaporated, hence long paths are chosen less and less. The implemented Ant Colony Optimization Based Algorithm (ACOBA) is based on the following principles:

• An ant starts path from a random node.

• Probability of moving toward another node depends on: (i) distance between current and new node, and (ii) intensity of pheromone on this path.

• After visiting all nodes, a solution Π^j is created. Afterward, the pheromone is updated and a new ant should start creating path.

The basic formula for $\tau_{ij}(t)$ – the pheromone intensity between vertices i and j, in some moment t, is expressed by (2):

$$\tau_{ij}(t+1) = \omega\,\tau_{ij}(t) + \Delta\tau_{ij} \qquad (2)$$

where $\Delta\tau_{ij} = mQ$ is the value of the change in pheromone intensity between nodes i and j caused by the number of m ants; Q is a basic value of the pheromone that is left by an ant between two nodes; ω is the decay factor.

Going to the next node is motivated by (i) the pheromone intensity, (ii) the list of just visited nodes, and (iii) the visibility of a node expressed by the formula $\eta_{ij} = 1/c_{ij}$, where c_{ij} is the distance between nodes i and j.

The formula of the probability that the ant can choose j-th node is expressed by (3):

$$p_{ij}^{k}(t) = [\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}] / \{\Sigma[\tau_{il}^{\alpha}(t)\eta_{il}^{\beta}]\},\ \text{if}\ \ j \in D_K \quad \text{or} \quad p_{ij}^{k}(t) = 0,\ \text{otherwise} \qquad (3)$$

where the summing Σ is being made for all l belonging to D_k. The parameters α and β can be chosen. High α value causes that ants are choosing mostly paths with pheromone, the small value causes that ACOBA behaves like a greedy algorithm.

3.3 SABA Algorithm

The classical idea of simulated annealing was proposed in [7]. An inspiration in developing was a technological process used in metallurgy – annealing. This process, described as thermal treatment, consists of three stages – preheat object to assumed temperature, soaking in this temperature and cooling with appropriate speed that can ensure thermal balance. The basic idea of SA is to generate a trajectory of searching a successive solutions Π^0, Π^1,(where each next solution Π^{j+1} is taken from some neighborhood of a current Π^j) in order to escape or avoid a local minimum. The last solution is considered as Π - the best found solution.

The Simulated Annealing Based Algorithm (SABA) starts working with some basic solution Π^0 (chosen at random) and successively is moving through adjacent solutions. In iteration i, from the neighborhood of a current solution Π^0, there is chosen randomly a perturbed solution Π'. Let $\Delta = F(\Pi') - F(\Pi^j)$, where F is expressed by (1). Then, two situations can occur: (a) $\Delta < 0$ or (b) $\Delta \geq 0$. In case (a), the perturbed solution is accepted as a new without any additional conditions, i.e., $\Pi^{j+1} = \Pi'$, in case (b) Π' is accepted with some probability $P = \{1, exp(-\Delta / T_i)\}$, where T_i is the temperature in i-th iteration, i.e.:

$$\Pi^{j+1} = \Pi' \Leftrightarrow R < exp(-\Delta / T_i) \qquad (4)$$

for some randomly generated $R \in [0, 1]$. Temperature T_i is the parameter that controls the search process and should be decreasing with the logarithmic cooling scheme:

$$T_{i+1} = T_i/(1+\lambda T_i), \quad \text{where} \quad \lambda = (T_0 - T_K) / (N T_0 T_K) \tag{5}$$

The parameters T_0 and T_k in (5) might be chosen experimentally: a very small T_k with the suggested value of 0.0001 and a very big T_0 with the suggested value of 10000. In each iteration *i*, the generation process of choosing a new solution is repeated *s* times for the temperature T_i, where *s* can be assumed as *n*, n^2, $n^2/2$, $n^2/4$, etc. The search procedure is terminated when the combination of the stopping criteria is fulfilled, i.e., $T_i = T_k$, or $i = MaxIter$ (the maximal number of iterations), or the procedure exceeds some assumed computational time.

4. Experimentation System

The experimentation system was created in Microsoft Visual Studio 2012 in C#. The main modules of the system are: Creating Graphs Module, Algorithms Implementation Module, and Visualization Module.

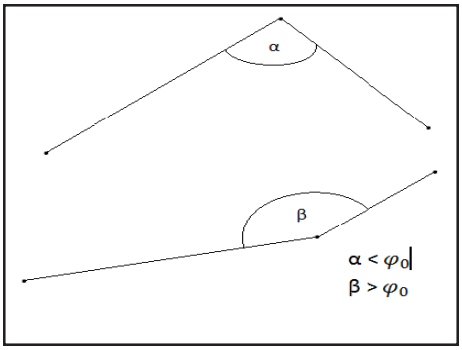

Figure 3. Analysis of angles

After completing operations an exemplary graph may look like in Figure 4 (the distances between nodes are in kilometers).

4.1 Creating Graphs Module

This module allows creating graph-map by activating three operations:

- *Connecting nodes 'smartly'* – The nodes are being connected taking into account the case that if between two nodes another (third) node is located then the position of the third node is analyzed - if the angle α or β angle (see Figure 3) is greater than some arbitrary j then these nodes will not be connected.
- *Removing intersections* – In this phase all the connections that are crossing each other are deleted (the mathematical methods are used to find the point of intersection).
- *Connecting remaining nodes* – After above described two operations, there is a possibility that some nodes would be unconnected with any other. Then these nodes would be connected with the nearest one.

4.2 Algorithms Implementation Module

There are four algorithms available, including NN, RS, SABA, and ACOBA. The construction of the computer programs allows developing the system and implementing other algorithms easily.

4.3 Visualization Module.

To visualize all routes produced by the algorithms, Gnuplot Environment was used for that purpose. The application ediates Gnuplot commands through pipes. An exemplary route obtained by ACOBA is presented in Figure 5.

4.4 Input-Output System

The experimentation system may be treated as the input – output system giving possibilities for making simulation experiments on the two stages. At the first stage, the various problem parameters can be changed and the effectiveness of the algorithms to solving problem formulated in Section 2 can be tested. At the second stage, the impact of the inner parameters of the algorithms can be observed in order to adjust their values to the categories of the problem.

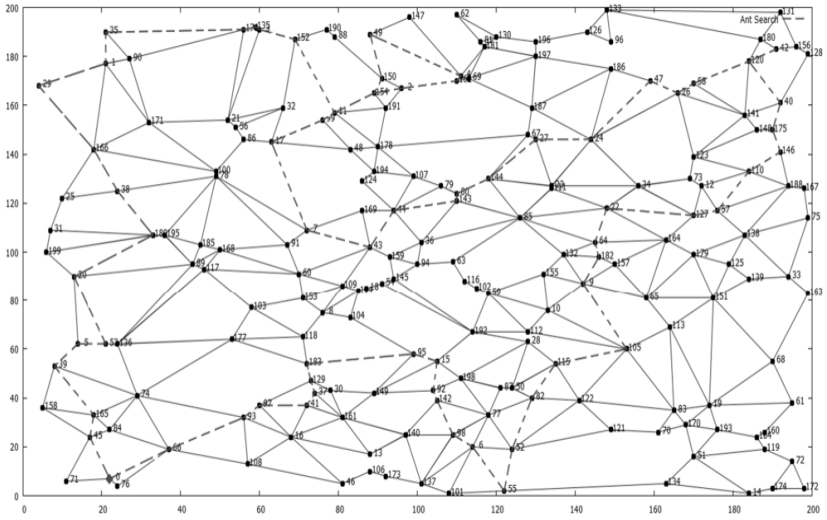

Figure 4. Exemplary graph

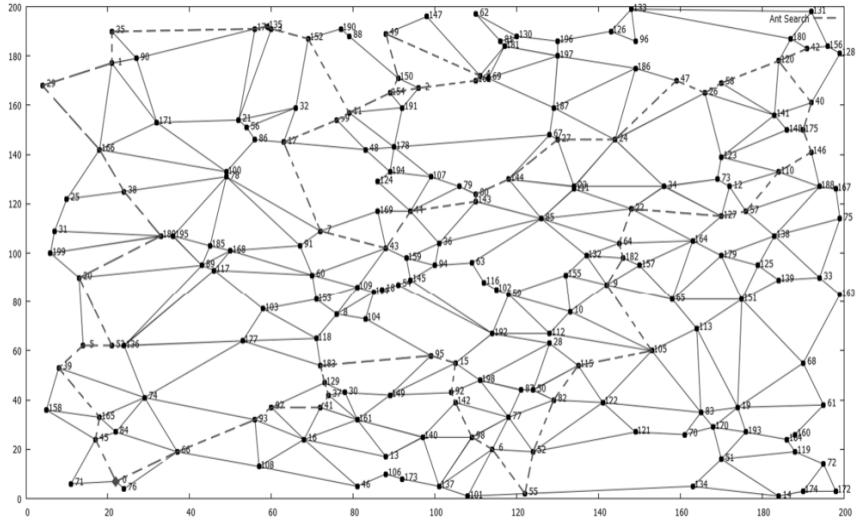

Figure 5. Visualization of exemplary solution

As inputs can be taken into consideration the following parameters:

Problem Parameters

The size of the graph – the variables n and N; matrices: T (times), D (distances), V (speeds); the weights a_1, and a_2 in the cost function (1); Fuel – the fuel consumption function.

Algorithms Parameters

For SABA: The temperatures T_0 and T_k; the parameter λ; the repetition parameter s; and *MaxIter*.
For ACOBA: The initial pheromone Q; the number of ants; the coefficients α, β, and ω.

As outputs can be taken:

- The resulting route Π, the total distance; the total traveling time; the average velocity; the fuel consumption; the average fuel consumption, and the following indices of performance:
- The value of the cost function $F(\Pi)$ defined by (1);
- The percentage improvement of the cost function (1) for a given algorithm in comparison to (1) obtained for *NN*;

• The computational time of the given algorithm.

5. Investigations

A lot of simulation experiments were carried out with the experimentation system. In this section the results of a complex experiment at the first stage are presented. The main goal of the experiment was the comparison between algorithms.

5.1 Experiment Design

The inner parameters for the metaheuristic algorithms were stated (in result of tuning after preliminary experiments) as:

For SABA: $T_0 = 10000$; $T_k = 0.0001$; $MaxIter = 5000$.

For ACOBA: $Q = 1$; the number of ants = $1000*n$; $\alpha = \beta = 1$, and $\omega = 0.5$.

All algorithms were tested on the following types of the graphs:

Case #1: $n = 10$, and $n = 20$ among 50 nodes available.

Case #2: $n = 20$, and $n = 40$ among 2000 nodes available.

Other inputs were like in examples shown in the previous sections. The results for any scenario were computed as the average values after 100 repetitions.

5.2 Results of Experiment

The most important results are presented in Table 1. The results are in the form of the values of the cost function (1) which were obtained for the four implemented algorithms in the both considered cases.

Total number of nodes		50		200	
n		10	20	20	40
Cost function $F(\Pi)$	RS	232.23	461.58	427.19	837.11
	NN	136.08	182.10	182.31	252.31
	SABA	131.88	179.70	181.44	252.31
	ACOBA	127.63	172.81	168.33	233.07

Table 1. The resulting cost function for the algorithms

In Table 2 the profits given by using the metaheuristic algorithms are shown. The profit was calculated as the percentage improvement of the cost function in comparison to the cost function obtained when NN was used.

Total number of nodes		50		200	
n		10	20	20	40
Improvement [%]	SABA	3.09	1.32	0.48	0.00
	ACOBA	6.21	5.10	7.67	7.62

Table 2. The percentage improvement given by metaheuristic algorithms

In Table 3 the comparison of the algorithms from the point of view of the total computational time needed for finding the solution to the considered problem is presented.

The results obtained from experiments made for one of the considered scenarios are presented in detail. All possible to consider outputs of the experimentation system are presented in Table 4, including the cost function, the traveling time, the distance ofthe founded route, and in particular the information about the fuel consumption.

Total number of nodes		50		200	
n		10	20	20	40
Cost function $F(\Pi)$	RS	2.01	2.10	2.08	2.20
	NN	1.05	1.06	1.10	1.31
	SABA	1094.10	2214.07	3349.51	7213.72
	ACOBA	546.05	3361.81	3684.12	21352.12

Table 3. The computational time of the algorithms

Algorithm	**NN**	**SABA**	**ACOBA**
Cost function	194.95	185.80	158.26
Distance [km]	1142.00	1038.00	908.00
Average velocity [km/h]	73.37	67.32	72.03
Traveling time [h]	15.56	15.42	12.61
Fuel consumption [l]	117.13	108.71	95.24
Average fuel consumption [l/km]	10.26	10.47	10.49

Table 4. Exemplary results provided by the algorithms

5.3 Comments

It may be observed from Tables 1 – 4 that:

- If graph is more complex, SABA provides worse results than ACOBA. ACOBA is very stable, in the meaning that improvement does not differ much in all cases.
- When the number of nodes that have to be visited is smaller, the both algorithms provide better results. There were cases, when ACOBA improved cost function up to 25%.
- The computational time for SABA is growing linearly (2 times more if the number of nodes increased twice), and ACOBA is also growing linearly, however, it grew 5 times while the number of the visited nodes was increased only 2 times.
- By analyzing Table 4 there can be noticed that ACOBA found over 200 km shorter route that this obtained by NN, and 100 km shorter than this obtained by SABA.
- What is also important, the fuel consumption in this particular case was up to 22 liters less and it caused that almost 3 hours was saved.

6. Conclusion

Two implemented metaheuristic algorithms SABA and ACOBA gave better results in minimizing the cost function than the NN and RS algorithms in any case. In particular, what was not expected, the highest profit was obtained by the algorithm based on ant colony optimization approach. Obviously, the metaheuristic algorithms needed more computational time than the simple NN and RS.

The main achievement of this work is the designed and implemented experimentation system which may give many opportunities for creating various scenarios for simulation experiments. One of the hardest aspects in implementing the system was creating a proper graph. Random disposing of points and random connecting them generate many intersections in graph. The proposed

procedure for creating a graph, described in Section 4, ensured to suit perfectly.

In the further research in the area, the authors are planning to consider more algorithms for solving the formulated environmental TSP, in particular algorithms based on other evolutionary approaches, e.g., presented in [9], and the hybrid algorithms described in [15], either.

There are also several interesting issues that might be considered in the future work in this area, as designing and implementing experimentation systems which can allow to conduct the multistage experiments in the automatic manners along with the ideas presented in [8].

Acknowledgement

This work was supported by the statutory funds of the Department of Systems and Computer Networks, Faculty of Electronics, Wroclaw University of Technology, Poland.

References

[1] Bonomi, E., Lutton, J. L. (1984). The N-city travelling salesman problem: *statistical mechanics and the Metropolis algorithm*. *SIAM Review*, 26 [4] , p. 551-568.

[2] Dorigo, M. (1992). Optimization, learning, and natural algorithms. PhD thesis, Politecnico di Milano, Italy.

[3] Feillet, D., Dejax, P., Gendreau, M. (2005). Traveling salesman problems with profits. *Transportation Science*, 39 [2] , p.188-205.

[4] Gutin, G., Punnen, A. P. (2002). The Traveling Salesman Problem and Its Variations. *Combinatorial Optimization*, Springer.

[5] Gutin, G., Yeo, A., Zverovich, A. (2002). Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP. *Discrete Applied Mathematics*.117 [1-3] , p. 81-86.

[6] Junger, M., Reinelt, G., Rinaldi, G. (1995). The traveling salesman problem. In: *Handbook in operation research and management sciences*, *North Holland*, Amsterdam, 7, p.225-330.

[7] Kirkpatrick, S., Gelatt, C. D. Jr., Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220, p. 671-680.

[8] Koszalka, L., Lisowski, D., Pozniak-Koszalka, I. (2006). Comparison of allocation algorithms for mesh structured networks using multistage simulation. *Lecture Notes in Computer Science*, Springer, 3984, p. 58-67.

[9] Ohia, D., Koszalka, L., Kasprzak, A. (2009). Evolutionary algorithm for solving congestion problem in computer network. *Lecture Notes in Computer Science*, Springer, 5711, p. 112-121.

[10] Ozceylan, E., Kiran, M. S., Atasagun, Y. (2014). A new hybrid heuristic approach for solving green traveling salesman problem. *In*: Proceedings of 41th Intern. Conf. on *Computers & Industrial Engineering*, p. 23-26.

[11] Saharidis, G. K. D., Kolomuos, G., Liberopoulos, G. (2014). Modeling and solution approach for the environmental TSP. *Engineering Letters*, 22, [2] , p.70-74.

[12] Solis, F. J., Wets, R. J-B. (1981). Minimization by random search techniques. *Journal Mathematics of Operation Research*. 6, [1] , p. 17-30.

[13] Suzuki, Y. (2011). A new truck routing approach for reducing fuel consumption and pollution emission. Transportation Research: *Transport and Environmental*. 16, [1] , p.73-77.

[14] Tasgetiren, F.M. and Smith, A.E. (2000). A genetic algorithm for the orienteering problem. *In*: Proceedings to Congress on Evolutionary Computations, San Diego, CA, p. 1190-1195.

[15] Tsai, C.F., Tsai, C.W., Tseng, C.C. (2004). A new hybrid heuristic approach for solving large traveling salesman problem. *Information Sciences.* 166, [1] , p. 23-26.