Modeling and Solution Approach for the Environmental Traveling Salesman Problem

Georgios K.D. Saharidis, George Kolomvos, George Liberopoulos

Abstract— **We consider the environmental traveling salesman problem in a connected graph driven by a cost function describing the impact of environmental externalities over the routes. The resulting problem is the asymmetric non-Euclidean TSP that we solve using a blend of cutting planes and 2-OPT algorithm. We test our solution approach on the well-known instances of the TSP-LIB and we present the results and the future research directions.**

Index Terms— **2-OPT, cutting planes, TSP.**

I. INTRODUCTION

In this paper the application is motivated by the environmental extension of the TSP, namely, to find the environmentally friendliest tour in a directed graph, the arcs of which are weighted based on their impact on fuel consumption. The environmental dimension of the problem has not been exhaustively addressed, although most countries and their governments do recognize the major effect of vehicle emissions on the environment.

The Green Vehicle Routing Problem was introduced in 2012 by Erdogan & Miller-Hooks in [16] where the vehicle driving range is dictated by fuel tank capacity limitations and tour duration constraints restrict tour durations to a pre-specified limit. The total distance traveled is still minimized. For a recent survey on the area the reader is referred to [17]. The novelty of our work lies on the fact that the criterion to be minimized is the environmental externalities score, briefly introduced in the following section, multiplied by the distance.

At the modeling side, the vast majority of works in the TSP is focused on minimizing the total distance of the tour. Although one may claim that fuel consumption is linked to the distance traveled, this may only be true should one assume that all routes are under identical conditions e.g. same quality of tarmac, same grade, identical wind speed and direction etc. We may encounter dozens of possible daily conditions where this assumption fails dramatically. It suffices to follow a route with large alteration in grade or strong side winds and the fuel consumption may increase at such level that a longer route with environmentally friendlier conditions would have been far more economical. The TSP may be amenable to the vehicle routing problem, consequently, the results of this paper may also be used to optimize the fleet management of light- or heavy-duty vehicles.

At the solution side, the TSP literature is vast. For a review of approaches to solve the TSP, the reader is referred to the comprehensive work of Laporte [8]. A more recent review with developments and an updated set of modern areas of applications is included in [2] and [12]. The description of the TSP polytope is not yet known and its complexity relies on the subtour elimination constraints. The formulation proposed by Dantzig, Fulekrson and Johnson in [4] provides a tight description of the polytope, but the number of constraints grows exponentially with the size of the problem, rendering the formulation impractical from the computational point of view. The idea that was firstly explored in [4] was to solve the LP relaxation and subsequently add cuts which are violated by the integer solutions and not by their continuous counterparts. This was the first cutting plane algorithm proposed in the TSP literature and was made popular thereof, since it showed to perform fairly well on the case of the 49-node problem addressed by the authors. Current algorithms [1] are able to attack problems of thousands of nodes, but still parallel computing is required to obtain an optimal solution within reasonable time.

Cutting planes algorithms can be divided into generic and structured cuts algorithms by the way one follows to determine which cuts they should append at each iteration. In the former case, one can base on algebraic arguments to generate cuts such as for instance Gomory cuts, lift-and-project or mixed-integer rounding cuts [3]. More emphasis in the literature is given on structured cuts, where the underlying structure of the specific problem is exploited to generate valid inequalities at each iteration. Consequently, one obtains 2-matching inequalities [5] and comb inequalities [6]. It is not surprising that these cuts are coming or may also be applied to other problems such as knapsack and vertex packing since the TSP is linked to these problems. These are commonly called in the literature as "brunch and cut" approaches, since one solves the relaxed version of the TSP where the integer constraints are dropped; apply valid inequalities; and continues with the branch and bound method. The work of Padberg & Rinaldi in [10] is considered as a milestone in the successful

Manuscript received November 29, 2013; revised December 23, 2013. This work was supported in part by the European Commission under the grant FP7-PEOPLE-2011-CIG, GreenRoute, 293753 and the grant EnvRouting SH3_(1234) of Action "Supporting Postdoctoral Researchers" of the Operational Program "Education and Lifelong Learning" (Action's Beneficiary: General Secretariat for Research and Technology, Greece), and is co-financed by the European Social Fund (ESF) and the Greek State.

Georgios K.D. Saharidis is at the Department of Mechanical Engineering, University of Thessaly, Volos, Greece. Tel.: +30-24210-74185; fax: +30-24210-74050. E-mail address: saharidis@gmail.com; saharidis@mie.uth.gr

George Kolomvos is at the Kathikas Institute of Research and Technology, Paphos, Cyprus.

George Liberopoulos is at the Department of Mechanical Engineering, University of Thessaly, Volos, Greece.

application of branch and cut strategies in the TSP. Branch and cut combine cutting planes with the well-known branch and bound algorithms according to which a smart enumeration of all different combinations is performed driven by the solution to the LP relaxation.

Another perspective of viewing the TSP is as a special case of a minimum 1-spanning tree. This analogy was nicely explored by Held & Karp in [15]. The idea is to carefully create an objective function such that the result of the spanning tree which is a lower bound of the TSP closely approximates the TSP. The formulation of the minimization of 1-spanning trees by default excludes subtours, so there is no reason to enforce any subtour elimination constraints. On the other hand, in a minimum spanning tree there may be nodes with a degree greater to two, that is for instance, a node with two descendants nodes, which is prohibited in the TSP.

Christofides algorithm [14] is based on the 1-spanning view perspective and provides a tight lower bound on the original TSP. The minimum spanning tree and the perfect matching problem are the two main operations performed, based on the triangular property or Euclidean graphs. In the literature the vast majority of applications and theory is devoted to the symmetric Euclidean TSP. In this paper we are interested in the asymmetric non-Euclidean TSPs.

Heuristics algorithms are largely employed to efficiently solve the TSP of larger instances. Heuristics are divided to construction heuristics and improvement heuristics. The former aim at constructing a tour from scratch usually including Euclidean arguments (such as the greedy algorithm for the nearest or k-th nearest neighbor, see for instance). The latter include methods that are given an initial tour and are trying to improve. Such algorithms are the 2-OPT algorithm that we apply in this paper, 3-OPT and k-OPT (known as LKH) proposed by Lin & Kernighanin [9] and successfully applied by Helsgaun in [7] to provide what is known today as the best generic algorithm for solving the TSP.

II. Modeling and solution approach

A. Environmental Externalities Score

Let us consider two different routing options for a vehicle to travel from a point A to a point B. The need to introduce the concept of the environmental externalities score arises when one wishes to compare these two routes that will be traveled by the same vehicle in terms of their environmental impact. In both cases, we may assume that the driving attitude will not change and of course that the vehicle characteristics will remain identical. In this work, we introduce the concept of the environmental externalities score, henceforth *EES*, which is a measure expressing the percentage of increase or decrease of the underlying environmental externalities compared to the nominal conditions. Each arc would have an individual *EES* based on the arc's characteristics. By multiplying the *EES* with the values provided by any emission calculation model, we may translate the result into fuel consumption in liters per kilometer.

In this paper we will not enter into the details of how the *EES* is calculated. The reader is referred to [13] for a thorough study on how this measure was conceived, devised and developed. We define the instantaneous environmental externalities score function *EES* related to fuel consumption to be the ratio of instantaneous fuel consumption to fuel consumption at nominal conditions. The above is expressed through the following formula: $EES = \frac{FC}{\overline{FC}}$ where FC stands for fuel consumption and $\overline{FC}$ for fuel consumption at nominal conditions. The consumption at nominal conditions is provided by an emission calculation model.

B. The Traveling Salesman problem

Let us now turn our attention towards the solution of the resulting environmental TSP. Let $G = (V, A)$ be a graph where V is a set of N vertices and A is a set of arcs. Let C be a cost matrix associated with A. V is the set of vertices such that $V = \{0,1,2,\dots,N\}$ and $i, j \in V$. We call the first vertex $i = 0$. Arcs connect vertices such that edge ij connects the vertices i and j. We denote by $x_{ij} \in \{0,1\}$ the binary variable which takes the value of 1 if the arc connecting i and j is included in the Hamiltonian tour and 0 if not. x is the vector containing the values x_{ij}. Let c_{ij} be the vector of costs associated to the edge ij. In fact c_{ij} is fed into the objective function using the values *EES* for every arc mentioned in the previous section. The formulation of the TSP without the subtour elimination constraints (SECs) is equivalent to the assignment problem (AP) and is presented right below.

$$(Assignment\ Problem: AP)$$

$$\min_{x_{ij}} c_{ij}x_{ij}$$

subject to:

$$\sum_{i \in V \setminus \{j\}} x_{ij} = 1, \forall j \in V$$

$$\sum_{j \in V \setminus \{i\}} x_{ij} = 1, \forall i \in V$$

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

The SECs can be represented in various ways following the formulations. In this paper we follow a cutting place approach, where the subtour elimination constraints are initially dropped. The resulting problem *(AP′)* is a relaxation of the original TSP in the sense that every feasible solution of the TSP is a feasible solution of *(AP′)*, while the optimal value of *(AP′)* provides a lower bound on the optimal value of the original TSP. Note that we call it *(AP′)*, because it is the assignment problem *(AP)* augmented with the cutting planes appended at each iteration.

At each iteration of our algorithm, we solve *(AP′)* and inspect the resulting solution x_{ij}. Note that we solve the *(AP′)* using the branch and bound method. If the resulting solution satisfies the SECs then we can safely claim that we have obtained the optimal path. If it does contain at least two subtours, then we have obtained a lower bound. We identify subtours by simple inspection. That is, we inspect the solution vector x_{ij} by starting from an arbitrary vertex (e.g. $i = 0$) keeping track of the path. If we reach the starting vertex without having visited all vertices then we have encountered a subtour. We enter the respective SEC and we proceed to identifying the next subtour.

We construct a feasible tour using the subtours identified earlier (we will explain later how we do this) and we apply a 2-OPT heuristic aiming to improve the incumbent solution. The role of the resulting 2-OPT feasible solution has a dual role: (a) to provide an upper bound at the current iteration and (b) to pass this information of the upper bound to the branch and bound method and accelerate branching by fathoming more nodes in the search tree. Moreover, using the subtours identified we generate a valid cut – explained below – and proceed to the next iteration.

Once all subtours identified and the respective valid inequalities entered, then we solve again *(AP′)* to optimality and repeat the procedure. The algorithm is presented in Table 1 as a pseudocode.

TABLE I
ALGORITHM IN PSEUDOCODE

Step 0:	(**Initialization**) Create the set $S = \emptyset$ that will contain the subtours identified. Set the iterator $k = 0$. Set lower bound $LB = \infty$ and upper bound $UB = -\infty$.
Step 1:	(**Solution of (AP')**) Solve (AP') to obtain the incumbent solution x_{ij}^k at iteration k. Store the value of the objective function at LB.
Step 2:	(**Subtour inspection**) Let $S_l \subset S$ be the subtour l which contains the vertices belonging to this subtour. Inspect the solution vector x to identify subtours S_l $l = 1,2$ ... until all vertices are inspected. Subtour inspection is performed starting from an arbitrary vertex and following connections. If no subtour is identified, i.e. $S = \emptyset$, then the incumbent solution x_{ij}^k is optimal representing a Hamiltonian tour and the algorithm ends here. Else continue to Step 3.
Step 3	(**Valid inequality**) For every subtour S_l, append the following cut at (AP'): $\sum_{i \in S_l, j \notin S_l} x_{ij} = 2y_l \; \forall \, S_l$ and $y_l \in \mathbb{N}^+$ *(1)*
Step 4:	(**Tour construction**) Consider the current subtour identified at Step 2. If this is the first subtour identified, then go to Step 2. Else, find the arc *ij* of the current tour with specific properties (see notes below) and place the identified subtour between *i* and *j*.
Step 5:	(**2-OPT**) Perform a 2-OPT move. If a 2-OPT move improves the *UB*, then (a) update the UB with the value of the new feasible tour and (b) set the UB as cut-off value for the branch and bound method and return to Step 1.

Let us explain a bit Steps 3, 4 and 5. In Step 3, the valid inequality (1) imposes an even number of connections between the nodes of the subtour and the remaining nodes not belonging to the subtour. Unless the incumbent solution of Step is optimal, it will include subtours. When a subtour is identified we instruct the algorithm to connect this subtour with the remaining vertices by an even number of arcs. For this reason we introduce the variable $y_l \in N$ for every identified subtour S_l, which denotes the number of arcs connecting the subtour with the remaining vertices. Because this number needs to be even, we actually use variable $2y_l$ and enforce constraint (1) for each subtour S_l identified.

The performance of the 2-OPT heuristic in Step 4 depends on the quality of the initial tour on which improvements will be tested. The idea exploited here is to feed the 2-OPT with as many candidate initial tours as possible and let it find the best improved tour among those. We have thus implemented several strategies, such as: (a) random positions of the current subtour in the current tour; (b) placing current subtour at the end of the current tour, (c) nearest neighbor algorithm; (d) the strategy that worked best in terms of results is the following: Every time we identify a subtour we place it between the two vertices inducing the most "expensive" pass of the tour. To illustrate this and for ease of explanation, let us represent the current tour as a string with starting point the node *0* and last node say *n* (in the tour, the last node *n* is also the preceding node of *0*). Iterate among all possible pairs of nodes *i* and $j \neq i$. Figure 1 illustrates this operation. All resulting tours are sent to the 2-OPT algorithm and an improved tour, if possible is returned. We keep the tour with the lowest upper bound.

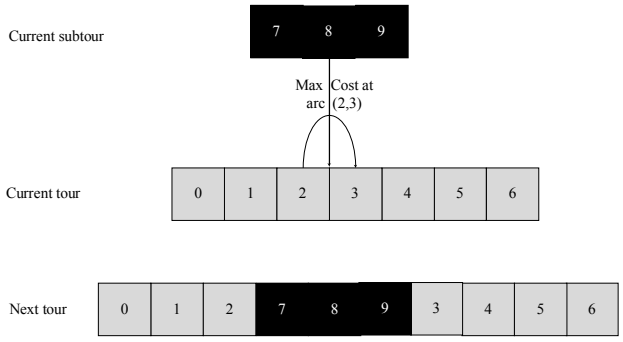


Fig. 1. Tour construction.

In Step 5, we perform a 2-OPT move. We adopt the representation of the string to illustrate this move. Name the node following i as i' and the node following j as j'. Select the pair *(i,j)* for which $c_{ii'} + c_{jj'} + c_{n0} > c_{ij'} + c_{ni'} + c_{j0}$ and perform a 2-OPT move by placing the string which is contained between the nodes i' and j, right after node n.

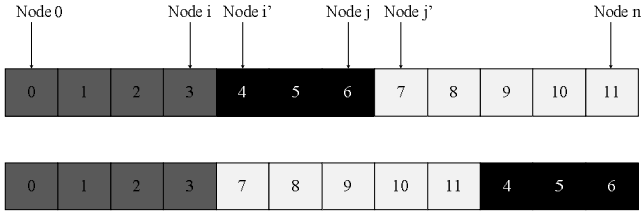


Fig. 2. 2-OPT move.

III. COMPUTATIONAL RESULTS

We report performance of our algorithm by testing it on the well-known TSP library. The TSP-LIB [11] is a library created by several TSP instances aimed to provide researchers with a broad set of test problems from various sources and with various properties. We have attacked problems included in the asymmetric test cases.

We performed the experiments on a dual-core 2.2GHz processor with 3GB of usable memory. The code was written on C++, the modeling of the problem was done using the IBM ILOG Concert Technology and the solution was provided by the IBM ILOG CPLEX 12.4 suite.

Table 4 presents the results of our runs. The column "*Name*" stores the name of the problem where the digits represent the number of nodes. The column "*#iterations*" stores the number of iteration until the end of the algorithm. The column "*CPU time*" stores the time required to reach the solution. We have divided the results in two pairs of columns: "*Optimal*" means that the algorithm run until the

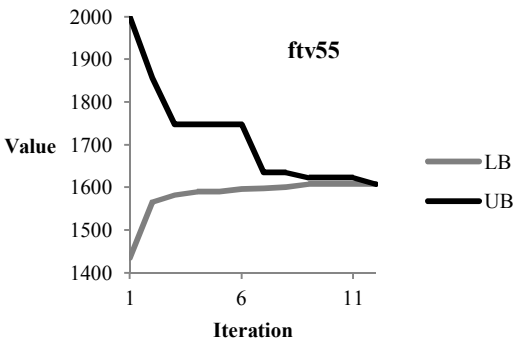
optimal solution was provided; "*Gap<1%*" means that the algorithm stopped once a feasible tour was obtained with a value close to optimum by at most 1%. We have included this number, because in the frame of the real-world application on which this algorithm will be applied, a 1% solution is considered as a satisfactory approximation.

TABLE II
RESULTS ON ASYMMETRIC INSTANCES OF THE TSP-LIB

	Optimal		Gap<1%	
Name	**#iterations**	**CPU time**	**#iterations**	**CPU time**
ftv64	12	5.679sec	9	4.15sec
ftv33	5	1.7sec	5	1.7sec
ftv35	11	2.543sec	10	2.403sec
ftv38	11	2.48sec	9	2.2sec
p43	16	7.816sec	3	1.498sec
ftv44	9	2.698sec	9	2.698sec
ftv47	10	3.573sec	10	3.573sec
ry48p	11	4.415sec	11	4.415sec
ft53	7	2.168sec	7	2.168sec
ftv55	12	4.181sec	9	3.26sec
ftv64	12	5.679sec	9	4.15sec
ft70	6	3.167sec	3	1.95sec
ftv70	7	4.93sec	7	4.93sec
kro124p	8	8.627sec	8	8.627sec
ftv170	11	46.441sec	11	46.441sec
rbg323	30	353.92sec	2	9.251sec
rbg358	70	1699.97sec	1	6.396sec
rbg403	1	7.489sec	1	7.489sec
rbg443	2	17.207sec	1	9.033sec

There may be cases where the 1%-solution is obtained relatively fast, while the optimum may take very long to reach. The most typical instance is rbg358 where the 1%-solution is obtained already from the first iteration, while the optimum is obtained at the end of the 70th iteration, almost half an hour later.

We generally observe that although LB starts with a value not very far from the optimum, it is actually UB that eventually decreases more rapidly. Figure 1 illustrates the convergence of the two bounds in the case of an instance with 55 nodes and another one with 124 nodes.



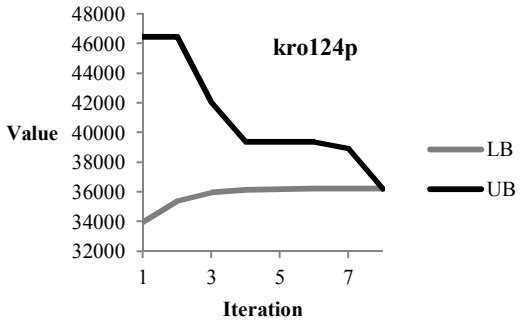


Fig. 3. Convergence of LB and UB for two instances.

The cuts passed to the (AP′) are dense which results to a heavy simplex tableau. A possible future research direction will be to seek ways to lower the density of the obtained cuts in order to find a compromise between sufficient information (high-density cuts) and compactness that will allow us attack larger instances (low-density cuts). Apart from a certain instances (ftv170), the CPU time obtained for each instance does not seem to increase rapidly with the number of nodes. Figure 2 illustrates this relation.

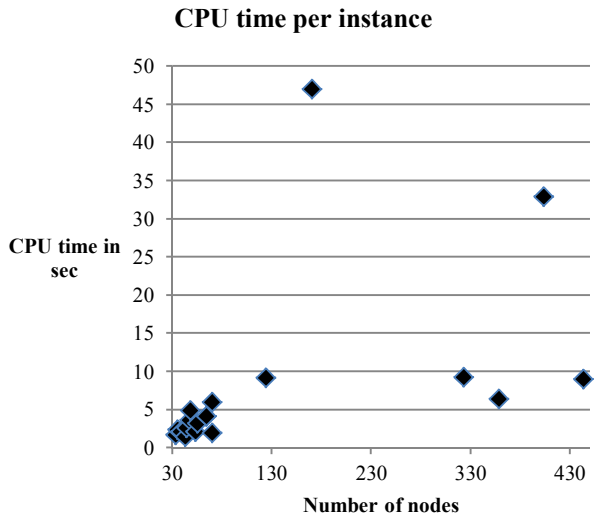


Fig. 4. Comparison of CPU time for different instances.

IV. CONCLUSION

In this paper we considered the environmental TSP, the first application of the TSP in an environmental dimension where the fuel consumption of the tour is minimized. We attacked in this paper two distinct fronts. We proposed a novel cost function the aim of which is to minimize the fuel consumed throughout a route. Two major limitations of emission calculation models have motivated this work: (a) they cannot cater for real-time conditions such as weather, traffic etc, the result being that they provide a rather myopic view of the actual fuel consumption when factors such as grade, weather and use of air condition come into play; and (b) they require substantial information on the type of vehicle, engine and either characteristics which is not straightforward to input in a web platform. The data we take into account are provided through freely available web APIs. The proposed EES reflects the increase or decrease of the nominal values (provided by any calculation model)

when a vehicle travels on a route segment. We based our model on available research and validated our results through comparison with an onboard diagnostic that we installed on a testing light-duty vehicle. The EES we devised feeds the objective function of the TSP.

The solution approach we followed for the TSP is based on the combination of a cutting plane strategy with 2-OPT heuristic. The cutting plane strategy appends a cut requiring an even number of connection between the nodes of each subtour and the remaining nodes outside the subtour. The resulting cuts are dense but effective. Solution of the assignment problem augmented with these valid inequalities provides an effective lower bound. The 2-OPT receives the different subtours and construct a feasible tour (not necessarily optimal however) and performs a 2-OPT move providing an upper bound. This upper bound is fed into the augmented problem when performing the branch and bound strategy to fathom nodes in the search tree. We tested the concept on the asymmetric instances of the TSP-LIB. The convergence of the proposed algorithm is reached after a few iterations, typically around 12. In the case one is interested in a solution within 1% margin of the optimum, then a tour is obtained typically within 10 seconds.

A future research direction will be to apply the environmental dimension of the TSP on heavy-duty vehicles and other transport modes. At the solution approach level, we will be investigating ways to make the appended cuts lighter (less dense) in order to benefit from economizing some computer memory.

REFERENCES

[1] Applegate, D., Bixby R., Chvatal V, Cook W (2001). TSP Cuts Which Do Not Conform to the Template Paradigm." In M Junger, D Naddef (eds.), *Computational Combinatorial Optimization, Lecture Notes In Computer Science*, Vol. 2241, pp. 261-304. Springer-Verlag Berlin Heidelberg.

[2] Bektas T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, Vol. 34, No. 3, pp. 209-219.

[3] Cornuejols, Gerard (2008). Valid Inequalities for Mixed Integer Linear Programs. *Mathematical Programming Ser. B*, 112:3-44.

[4] Dantzig, G.B., Fulkerson, D.R. and Johnson, S.M. (1954). Solution of a large scale traveling-salesman problem. *Operations Research,* Vol. 2, pp. 393-410.

[5] Edmonds, J. (1965). Maximum matching and a polyhedron with 0–1 vertices. *J. Res. Nat. Bur. Standards* 69B 125–130.

[6] Grötschel M. and Padberg M.W. (1979). On the symmetric travelling salesman problem I: Inequalities, *Mathematical Programming*, Volume 16, Issue 1, pp 265-280.

[7] Helsgaun, K., (2000). "An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic". *European Journal of Operational Research* 126 (1): 106–130. doi:10.1016/S0377-2217(99)00284-2.

[8] Laporte G. (1992). The Traveling Salesman Problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59, 1992, pp. 231-247.

[9] Lin, S. & Kernighan, B. W. (1973). "An Effective Heuristic Algorithm for the Traveling-Salesman Problem". *Operations Research* 21 (2): 498–516. doi:10.1287/opre.21.2.498.

[10] Padberg, M.W., Rinaldi, G., (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems, *SIAM Rev*. 33 pp 60–100.

[11] Reinelt, G., (1991). TSPLIB - A Traveling Salesman Problem Library. *INFORMS Journal on Computing,* 01/1991, 3:376-384.

[12] Saharidis G.K.D. (2014). Review of solution approaches for the symmetric traveling salesman problem. *International Journal of Information Systems and Supply Chain Management.* 2014, unpublished.

[13] Saharidis, G. (2012). Research report that presents the candidate TN factors for the definition of EESarc and the revision methodology following the development of new emission calculation models. FP7-PEOPLE-2011-CIG, GreenRoute: A web based platform which helps individuals and companies move commodities with the most environmental friendly way, minimizing emissions and transportation cost. Athens: Marie-Curie Grant.

[14] Christofides, N. (1976). Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University.

[15] Held, M, and Karp, R.M. (1969). The traveling-salesman problem and minimum spanning trees. New York : IBM Systems Research Institute.

[16] Canhong Lin, K.L. Choy, G.T.S. Ho, S.H. Chung, H.Y. Lam, Survey of Green Vehicle Routing Problem: Past and future trends, Expert Systems with Applications, Vol. 41, Issue 4, Part 1, March 2014, Pages 1118–1138.

[17] Sevgi Erdogan, Elise Miller-Hooks, A Green Vehicle Routing Problem, Transportation Research Part E 48 (2012) 100–114.