Applied Mathematics, 2014, 5, 2764-2770
Published Online October 2014 in SciRes. http://www.scirp.org/journal/am
http://dx.doi.org/10.4236/am.2014.517264

Scientific Research

The Algorithm of the Time-Dependent Shortest Path Problem with Time Windows

Nasser A. El-Sherbeny[1,2]

[1]Mathematics Department, Faculty of Science, Al-Azhar University, Cairo, Egypt
[2]Mathematics Department, Faculty of Applied Medical Science, Taif University, Turabah, KSA
Email: nasserelsherbeny@yahoo.com

Received 15 August 2014; revised 2 September 2014; accepted 9 September 2014

Abstract

In this paper, we present a new algorithm of the time-dependent shortest path problem with time windows. Give a directed graph $G=(V,E)$, where V is a set of nodes, E is a set of edges with a non-negative transit-time function $c_e(t)$. For each node $v\in V$, a time window $[a_v,b_v]$ within which the node may be visited and $a_v\le t\le b_v$, $t\in T$ is non-negative of the service and leaving time of the node. A source node s, a destination node d and a departure time t_0, the time-dependent shortest path problem with time windows asks to find an s,d-path that leaves a source node s at a departure time t_0; and minimizes the total arrival time at a destination node d. This formulation generalizes the classical shortest path problem in which c_e are constants. Our algorithm of the time windows gave the generalization of the ALT algorithm and A^* algorithm for the classical problem according to Goldberg and Harrelson [1], Dreyfus [2] and Hart *et al*. [3].

Keywords

Shortest Path, Time-Dependent Shortest Path, ALT Algorithm, A^* Algorithm, Time Windows

1. Introduction

The shortest path problem on graphs is a problem with many real-life applications such as: route planning in an internet, car navigation system, traffic simulation or logistic optimization. The shortest path problem is a classical combinatorial optimization problem. It has countless applications and so far numerous algorithms have been proposed (see Ahuja *et al*. [4]) including the well-known Dijkstra's algorithm. Recently, because some of the new improvement becomes fairly difficult, researchers began to study variants of this problem which include the time-dependent and the time windows generalization.

How to cite this paper: El-Sherbeny, N.A. (2014) The Algorithm of the Time-Dependent Shortest Path Problem with Time Windows. *Applied Mathematics*, **5**, 2764-2770. http://dx.doi.org/10.4236/am.2014.517264

Give a directed graph $G=(V,E)$, where V is a set of nodes, E is a set of edges, $c_e(t)$ is a non-negative transit-time function. For each node $v \in V$, a time window $[a_v, b_v]$ within which the node may be visited and $a_v \le t \le b_v$, $t \in T$ is non-negative of the service and leaving time of the node. A source node $s \in V$ with time window $[a_s, b_s]$, a destination node $d \in V$ with time window $[a_d, b_d]$, and a departure time t_o. The time-dependent shortest path problem with time windows asks to find an *s*, *d*-path that leaves a source node *s* at time t_o and minimizes the total arrival time at a destination node *d* which satisfies the set of all constraints (see El-Sherbeny [5], El-Sherbeny and Tuyttens [6], Tuyttens *et al*. [7] and El-Sherbeny [8]). One can notice that the undirected graphs can be treated by replacing each undirected edge with two reverse directed edges. Without losing of the generalization, we suppose that a destination node *d* is reachable from a source node *s*. For simplicity, we suppose that the domain of the definition for all $c_e(t)$ is $\Re^+$, but our algorithms work for the discrete version too. We also assume the time complexity to calculate a $c_e(t)$ which is bounded by some constant α. This formulation generalizes the classical shortest path problem with constant $c_e(t)$ and t_o. It can further handle time-variable edge costs, thus it has more application than the classical one, which is also referred to as the static problem in contrast.

In Cook and Halsey [9], it has considered and given a dynamic programming algorithm which is not polynomial-time at all. Dreyfus [2] suggested a polynomial-time straightforward generalization of the Dijkstra's algorithm. However, he did not notice that it works correctly only for instances satisfying the First-In First-Out (FIFO) property, *i.e*., for any edge $e=(v,w) \in E$ and $t_v \le t_w$, it holds that $t_v + c_e(t_v) \le t_w + c_e(t_w)$. In other words, the arrival-time function $t + c_e(t)$ is non-decreasing. With this property, we can ensure that there is no cycle of negative transit-time, hence a simple optimal solution exists. This was pointed out and discussed later (see Halpern [10], Kaufman and Smith [11] and Orda and Rom [12]).

On the other hand, the general problem without the FIFO constraint is NP-hard if the waiting at nodes is not allowed (see Sherali *et al*. [13]). In Orda and Rom [12], it showed that, if the waiting at nodes is allowed, which is natural in transportation systems, any instance can be converted to an equivalent instance that satisfies the FIFO property; hence, no waiting is needed, and that can be done in polynomial time (if $c_e(t)$ can be calculated in polynomial time). Thus, in the following, we will only consider instances that satisfy the FIFO property.

Even with the FIFO constraint, unlike the static case, studies are not rich. Dreyfus's proposal of the generalized Dijkstra's algorithm, despite of many studies (see Dean [14], Ding *et al*. [15], Halpern [10], Kanoulas *et al*. [16], Kaufman and Smith [11] and Orda and Rom [12]), there was no significant advancement in solving the problem more efficiently.

In this paper, we give a new algorithm of the time-dependent shortest path problem with time windows that generalizes the ALT algorithm (see Goldberg and Harralson [1]) and A^* algorithm for the static problem, unlike the generalized Dijkstra's algorithm, which uses a function *h* to estimate the distances between nodes in the graph in Section 2. In Section 3, we give an application instance of our algorithm and a generalization of the ALT algorithm (see Goldberg and Harralson [1], Dreyfus [2] and Hart *et al*. [3]) that is based on the static A^* algorithm and is faster than the Dijkstra's algorithm using preprocessing. Thus, we have found the first algorithm for the time-dependent shortest path problem with time windows that speeds up the calculation using preprocessing and we have observed that it is several time faster than the generalized Dijkstra's algorithm. Finally, the conclusion is given in Section 4.

2. The Algorithm of the Time-Dependent Shortest Path Problem with Time Windows

We start from the classical and well-known Dijkstra's algorithm. For each edge $e=(v,w) \in E$, we suppose that $c_e(t)=c_e$ is a constant. The service and leaving time to node $v \in V$ is $t \in T$ and $[a_v, b_v]$ a time window where, $a_v \le t \le b_v$. If $t_o = 0$, the Dijkstra's algorithm tries to find a shortest s, d -path in greedy manner. Let $p(v)$ denote the precedent node of a node *v* of the shortest *s*, *v*-path found so far. The Dijkstra's algorithm maintains for each node *v* a status $(v) \in$ {"unlabeled", "labeled", "finished"} and a distance label $g(v)$. At the beginning, $g(s)$ is the set to 0 and all status (v) is initialized to "unlabeled" except that *s* is "labeled". Then it repeatedly find a "labeled" node *v* with the smallest $g(v)$ (such *v* is called the active node) until $v = d$; then it tries to relax all non "finished" neighbors *w* of *v*, *i.e*., if status $(w) =$ "unlabeled" then the set it to "labeled" and let $g(w) = g(v) + c_{(v,w)}$, $p(w) = v$; otherwise status $(w) =$ "labeled". The time window of the node $w \in V$ is $[a_w, b_w]$, where $a_w \le t \le b_w$, $t \in T$. Let $g(w) = g(v) + c_{(v,w)}$, $p(w) = v$ if $g(w) \succ g(v) + c_{(v,w)}$;

after all these have done, set status (v) to "finished" and continue. See **Table 1** for the pseudo-code.

The our algorithm of the A^* algorithm given in (**Table 2**) follows the same fashion except that it employs an estimator $h(v)$ for all $v \in V$ with the time window $[a_v, b_v]$ and chooses the active node by the smallest $g(v)+h(v)$. A good estimator $h(v)$ for all $v \in V$ can be used to reduce the search space (*i.e.* the set of nodes that have to be explored before the solution is found) of the shortest path queries effectively. Notice that how to determine $h(v)$ is not part of the algorithm. It must be obtained by some other method, and the choice of h determines the correctness and the efficiency of the A^* algorithm (a good lower-bound on the v, d -distance is preferred). Clearly the Dijkstra's algorithm is a special case with $h=0$.

Remark: The Dijkstra's algorithm is a special case of $h=0$. For general h, however, the correctness is not guaranteed.

Now we are ready to describe our generalized A^* algorithm. It generalizes $h(v)$ by the time dependent version $h(v,t)$ with $[a_v, b_v]$ is the time windows of a node v where $a_v \le t \le b_v$, $t \in T$ is the service and leaving time of the node. Thus in **Table 3**, we use $h(v, g(v))$ to replace $h(v)$. Notice the rule for choosing the active node (Line 2) has been changed in addition.

Definition 2.1. Given a directed graph $G=(V,E)$, a non-negative transit-time function $c_e(t)$ of each edge $e=(v,w) \in E$, and $[a_v, b_v]$, is a time windows, $a_v \le t \le b_v$, $t \in T$ is the service and leaving time to node v, then for all edges $h(v,t) \le c_e(t) + h(w, t+c_e(t))$ is called a triangle condition.

In a directed graph $G=(V,E)$, a non-negative transit-time function $c_e(t)$ of each edge $e=(v,w) \in E$, with $[a_v, b_v]$ is a time windows, $a_v \le t \le b_v$, $t \in T$ is the service and leaving time to node v, a source node s, a destination node d and a departure time t_0 at a source node s of the time-dependent shortest path problem with time windows such that the FIFO properly is satisfies and d is reachable from s, the generalized of A^* algorithm in **Table 3** finds an optimal solution if h satisfies the three conditions:

- For all vertices v, $w \in V$ and $t_v \le t_w$, $t_v + h(v,t_v) \le t_w + h(w,t_w)$ is the FIFO time windows condition (2.1).
- For all edges $e=(v,w) \in E$ and $t \in T$, $h(v,t) \le c_e(t) + h(w, t+c_e(t))$ is a triangle condition (2.2).
- For all vertices v, $w \in V$ and $t_v \le t_w$, $[a_v, b_v] \le [a_w, b_w]$ is the time windows condition (2.3).

Table 1. Pseudo-code of the Dijkstra's algorithm for the static shortest path problem time windows.

1) status (s):= "labeled", $g(s):=0$, status (v):= "unlabeled" for all $v \ne s$
2) Let v be a "labeled" node with the time window (*i.e.*, $[a_v, b_v]$, $a_v \le t \le b_v$, $t \in T$) and the smallest $g(v)$ (the active node). IF $v=d$ GOTO 11)
3) FOR all edges $(v,w) \in E$ DO
4) IF status $(w)=$ "unlabeled" THEN
5) status (w):= "labeled" with the time window (*i.e.*, $[a_w, b_w]$, $a_w \le t \le b_w$, $t \in T$), $g(w):=g(v)+c_{(v,w)}$, $p(w):=v$
6) ELSE IF status $(w)=$ "labeled" AND $g(w) \succ g(v)+c_{(v,w)}$ THEN
7) $g(w):=g(v)+c_{(v,w)}$, $p(w):=v$
8) END IF
9) DONE
10) status (v):= "finished". GOTO 2)
11) OUTPUT $g(d)$ and the s, d -path found with $[a_s, b_s]$, $[a_d, b_d]$, $a_d \le t \le b_d$, $t \in T$ are the time windows of the source node s and a destination node d respectively (*i.e.* the reverse of $d, p(d), p(p(d)), \cdots, s$).

Table 2. Pseudo-code of the A^* algorithm for the static problem time windows.

Table 1
2) Let v be a "labeled" with time window $[a_v, b_v]$, $a_v \le t \le b_v$, $t \in T$ and the smallest $g(v)+h(v)$. IF $v=d$ GOTO 11)
Table 1

Table 3. Pseudo-code of A^* algorithm for the time-dependent shortest path problem with time windows.

1) Status $(s) :=$ "labeled", $g(s) := t_0$, status $(v) :=$ "unlabeled" for all $v \neq s$
2) Let v be a "labeled" node with time window $[a_v, b_v]$, $a_v \le t \le b_v$, and $t \in T$ is the service and leaving time at node v, the smallest $g(v) + h(v, g(v))$. In the case that there are multiple candidates, choose one with the smallest $g(v)$. IF $v = d$ GOTO 11)
3) FOR all edges $(v, w) \in E$ DO
4) IF status (w) is "unlabeled" THEN
5) status $(w) :=$ "labeled", $g(w) := g(v) + c_{(v,w)}(g(v))$, with time windows $[a_w, b_w]$, $a_w \le t \le b_w$, $t \in T$, $p(w) := v$
6) ELSE IF status (w) is "labeled" AND $g(w) \succ g(v) + c_{(v,w)}(g(v))$ THEN
7) $g(w) := g(v) + c_{(v,w)}(g(v))$, $p(w) := v$
8) END IF
9) DONE
10) status $(v) :=$ "finished". GOTO 2)
11) OUTPUT $g(d)$ and the s, d-path found with $[a_s, b_s]$, $[a_d, b_d]$, $a_d \le t \le b_d$, $t \in T$ are the time windows of the source node s and a destination node d respectively (*i.e.* the reverse of $d, p(d), p(p(d)), \cdots, s$).

The triangle condition (2.2) (see **Figure 1**) is a natural generalization from the classical A^* algorithm whereas the FIFO condition is only available in the time-dependent and time windows case. The generalized Dijkstra's algorithm is nothing but the simplest case with $h = 0$, and the generalization of Kanoulas *et al.* [16], on the other hand, simply uses a constant function $h(v,t) = h(v)$, with the time windows $[a_v, b_v]$ and $a_v \le t \le b_v$, $t \in T$ is the service and leaving time to a node $v \in V$ thus, it also a simple special-case of our algorithm.

Roughly speaking, it says the supposed transit-time $h(v,t)$ from v to d is no more than $c_e(t) + h(w, t + c_e(t))$, *i.e.* the supposed transit-time of the v, d-path $v \to w \to d$. Notice that, $h(w, t + c_e(t))$ is the supposed transit-time from w to d by leaving w at time $t + c_e(t)$ with the time windows $[a_w, b_w]$ and $t_w \le t \le b_w$, $t \in T$.

Theorem 2.1. Let $p = v_1, v_2, \cdots, v_k$ be a path with the time windows $[a_{v_1}, b_{v_1}]$ and $a_{v_1} \le t \le b_{v_1}$, $t \in T$ is the service and leaving time at node v_1. Define $\sigma_1 = 0$ and $\sigma_i = \sum_{j=1}^{i-1} c_{(v_j, v_{j+1})}(t + \sigma_j)$ be the transit-time from v_1 to v_i, $i = 2, \cdots, k$. Then it holds that $h(v_1, t) \le \sigma_k + h(v_k, t + \sigma_k)$.

Proof. By the above conditions (2.1), (2.2) and (2.3). We show by the induction that, every active node v must get the optimal distance label (the induction variable is the number of nodes in the shortest path), *i.e.*, the earliest arrival time at node v for leaving s at time t_0.

Let v be an active node satisfies the time windows $[a_v, b_v]$ and $a_v \le t \le b_v$, $t \in T$ is the service and leaving time of this node. If $v = s$, we are done. Otherwise, let p be a simple optimal s, v-path (it exists) and w be the first node on p such that status $(w) \neq$ "finished". Clearly w must exist and $w \neq s$ (it can be v) see **Figure 2**.

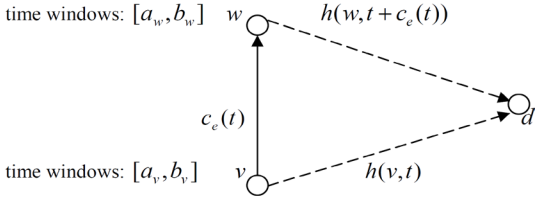

Figure 1. The triangle condition with time windows for the function h.

Figure 2. An optimal s, v-path with time windows is being considered. s, u: finished nodes; w: the first non-finished node; v: the active node.

Let g^* denote the optimal distance (*i.e*. the earliest arrival time). It is obvious that $g(w)=g^*(w)$ because w was relaxed when the precedent node u of w was active and at that time $g(u)=g^*(u)$ by the induction hypothesis. Let $\sigma=g^*(v)-g^*(w)$ be the shortest transit-time from w to v at departure time $g^*(w)$ (notice $\sigma\geq 0$). By applying the above conditions (2.1), (2.2) and (2.3) to the w, v-path with time windows on p with $t=g^*(w)$ we have

$$h\left(w,g^*(w)\right)\leq\sigma+h\left(v,g^*(w)+\sigma\right)=g^*(v)-g^*(w)+h\left(g^*(v)\right) \tag{2.4}$$

That is equivalent to

$$g^*(w)+h\left(w,g^*(w)\right)\leq g^*(v)+h\left(w,g^*(v)\right) \tag{2.5}$$

Then, since v is the active node with the time windows $[a_v,b_v]$ (thus has the smallest $g(v)+h(v,g(v))$) we have

$$g(v)+h\left(v,g(v)\right)\leq g(w)+h\left(w,g(w)\right)=g^*(w)+h\left(w,g^*(w)\right)\leq g^*(v)+h\left(w,g^*(v)\right) \tag{2.6}$$

On the other hand, by the FIFO condition and $g^*(v)\leq g(v)$ (the optimality of g^*), we have

$$g^*(v)+h\left(v,g^*(v)\right)\leq g(v)+h\left(v,g(v)\right) \tag{2.7}$$

Therefore we get the next fact by combining (2.6) and (2.7), we get

$$g(v)+h\left(v,g(v)\right)\leq g(w)+h\left(w,g(w)\right)=g^*(w)+h\left(w,g^*(w)\right)\leq g(v)+h\left(v,g(v)\right) \tag{2.8}$$

This means the equalities hold, hence $g(v)+h\left(v,g(v)\right)=g(w)+h\left(w,g(w)\right)$. Then by our choice of the active node, $g(v)\leq g(w)$ must hold. Thus $g(v)\leq g^*(w)\leq g^*(v)$ hence $g(v)=g^*(v)$.

Remark: The analogously to the static version, an h with $h(d,t)=0$ implies $h(v,t)$ where v satisfies the time windows $[a_v,b_v]$ and $a_v\leq t\leq b_v$, $t\in T$ is a lower bound on the shortest transit-time from v to d with leaving time $g(v)$ (by Theorem 2.1). Moreover, it is not difficult to show that with an h satisfying $h(d,t)=0$ and $h\geq 0$, the search space (the set of active nodes) of the generalized A^* algorithm is no longer than that the generalized Dijkstra's algorithm. Using this observation, we will give our algorithm in the next section that is practically faster than the generalized Dijkstra's algorithm.

3. Application Instance

The time complexity of the generalized Dijkstra's algorithm is $O(n\log n+m\alpha)$ by using a Fibonacci heap (we note it was $(O(m+n\log n)\alpha)$ in (Ding *et al*. [15]), where m, n, α are the number of edges, the number of nodes, and the time complexity to calculate $c_e(t)$, respectively. While we cannot improve this theoretical bound, let us give a practically faster algorithm that is based on our A^* algorithm and generalizes the static landmark-based ALT algorithm (Goldberg and Harrelsin [1], Dreyfus [2], and Hart *et al*. [3]).

The ALT algorithm is such as an algorithm that is supposed to answer the shortest-path queries for a known graph. This means we can preprocess the graph beforehand and use it to answer a query faster than a normal calculation by the Dijkstra's algorithm. Of course there is a trivial method of saving solutions for all possible queries and answers a query in $O(1)$ time, but the n^2 order (for the static case) is big (if not impossible) for large graphs, usually a road network is spares (*i.e*., $m\leq kn$ for some small k) and has several millions of nodes. So researchers are seeking efficient algorithm that uses $O(n)$ storage, see Wagnar and Willhalm [17] for a review. While this is an extremely hot topic for the static problem of these several years, for the time-dependent case, as far as we know, there was no proposal before our work.

Now let us describe the detail of our generalized ALT algorithm. Let $\tau^*(v,w,t)$ denote the shortest transit-time from a node v with the time windows $[a_v,b_v]$ to anther node w with a time windows $[a_w,b_w]$, a service and leaving time $t\in T$, $a_w\leq t\leq b_w$, hence we want to find an s, d-path of transit-time $\tau^*(v,w,t_0)$ Suppose we have a node z with time windows $[a_z,b_z]$ and the values $\tau^*(z,v,t)$ for all nodes v and all t (z is called a landmark). Also, suppose we can calculate a $\hat{t}$ (if exists) that

$$\hat{t}=\max\left\{t':t'+\tau^*(z,v,t')\leq t\right\} \tag{3.1}$$

In other words, $\hat{t}$ is the latest leaving time in order to get v before t (from z). Define h by:

$$h_z(v,t) = \max\left\{\tau^*(z,d,\hat{t}) - \tau^*(z,v,\hat{t}), 0\right\} \text{ if } \hat{t} \text{ exists, 0 otherwise (}i.e.,\ \hat{t} \text{ does not exist)} \quad (3.2)$$

It is clear that $h_z(d,t) = 0$ and $h_z \geq 0$. Actually this definition is a generalization from the static case, *i.e.*, h_z is an estimation (a lower bound) on the v, d transit-time, which is no shorter than the right side of (3.2) (by the triangle inequality due to the optimality of τ^*). Moreover, we can show that h_z satisfies the FIFO condition, the triangle condition and the time windows condition at the same time, too. The proof is not trivial nor difficult, but due to the page limit, we omit it in this work. We note it is important to choose $\hat{t}$ to be the maximum.

We still have to show how to calculate $\hat{t}$, which usually is difficult if there is no explicit expression for $\tau^*(z,v,t)$. Moreover, in general it is difficult to hold all values of $\tau^*(z,v,t)$. Fortunately, however, we can show that sampling of time works, *i.e.*, we can calculate and hold values $\tau^*(z,v,t_i)$ only for some $t_1 \prec t_2 \prec \cdots \prec t_k$ and define $\hat{t}$, if it exists, by

$$\hat{t} = \max\left\{t_i : t_i + \tau^*(z,v,t_i) \leq t\right\} \quad (3.3)$$

Again, we can show the function h_z defined by (3.2) with the above $\hat{t}$ satisfies the FIFO conditions, the triangle condition, the time windows condition, and $h_z(d,t) = 0$, $h_z \geq 0$. Moreover, we can employ more than one land marks to get a better estimation (notice the maximum of all $h_z s$ works). Applying this generalized ALT algorithm to a number of US road networks (obtained from the web site of the 9th DIMACS implementation challenge http://www.dis.uniromal.it/~challenge9/, where $320,000 \leq n \leq 1,210,000$ and $(m \leq 3n)$ with periodic piecewise-linear transit-time functions (with 9 samples a day), we have noticed that it ran at an average of about 4 times faster than the generalized Dijkstra's algorithm with 16 landmarks and 2 time samplings.

A comparison example of the search space between the generalized Dijkstra's algorithm and the generalized ALT algorithm for the time dependent shortest path problem time windows and our ALT algorithm for an instance with the number of nodes are 321,270 and the number of edges are 800,172. The number of landmarks is 16 and the number of time samplings is 2. The search space of the ALT algorithm is 0.055 smaller and the running time is 7.4 times faster.

4. Conclusion

In this paper, we present a new algorithm framework of A^* algorithm for the time-dependent shortest path problem with time windows. By constructing some appropriate estimator h, it is possible to get an algorithm that is faster than a normal generalized Dijkstra's algorithm. As an example, we have generalized the landmark based ALT algorithm, which we believe is the first algorithm that uses preprocessing to speed up the calculation of time-dependent shortest paths problem with time windows. Our experimental result shows that it is several times faster than a normal generalized Dijkstra's algorithm for large road networks.

Acknowledgements

The author would like to thank an anonymous referee for some useful comments.

References

[1] Goldberg, A. and Harrelson, C. (2005) Computing the Shortest Path: A^* Search Meets Graph Theory. http://research.microsoft.com/pubs/154937/soda05.pdf

[2] Dreyfus, S. (1969) An Appraisal of Some Shortest-Path Algorithms. *Operations Research*, **17**, 395-412. http://dx.doi.org/10.1287/opre.17.3.395

[3] Hart, P., Nilsson, N. and Raphael, B. (1968) A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions Systems Science and Cybernetics*, **4**, 100-107. http://dx.doi.org/10.1109/TSSC.1968.300136

[4] Ahuja, R., Magnanti, T. and Orlin, J. (1993) Network Flows: Theory, Algorithms, and Applications. Prentice-Hall, Upper Saddle River.

[5] El-Sherbeny, N. (2001) Resolution of a Vehicle Routing Problem with Multiobjective Simulated Annealing Method. Ph.D. Dissertation of Faculty of Science, Mons University, Mons.

[6] El-Sherbeny, N. and Tuyttens, D. (2001) Optimization Multicriteria of Routing Problem. Troisieme Journee de Travail sur la Programming Mathematique Multi-Objective, Faculte Polytechnique de Mons, Mons.

[7] Tuyttens, D., Teghem, J. and El-Sherbeny, N. (2004) A Particular Multiobjective Vehicle Routing Problem Solved by Simulated Annealing. *Lecture Notes in Economics and Mathematical Systems*, **535**, 133-152. http://dx.doi.org/10.1007/978-3-642-17144-4_5

[8] El-Sherbeny, N. (2011) Imprecision and Flexible Constraints in Fuzzy Vehicle Routing Problem. *American Journal of Mathematical and Management Sciences*, **31**, 55-71. http://dx.doi.org/10.1080/01966324.2011.10737800

[9] Cook, K. and Halsey, E. (1966) The Shortest Route through a Network with Time-Dependent Intermodal Transit. *Journal of Mathematical Analysis and Applications*, **14**, 493-498. http://dx.doi.org/10.1016/0022-247X(66)90009-6

[10] Halpern, H. (1977) Shortest Route with Time Dependent Length of Edges and Limited Delay Possibilities in Nodes. Operations *Research*, **21**, 117-124.

[11] Kaufman, D. and Smith, R. (1993) Fastest Paths in Time-Dependent Networks for Intelligent Vehicle-Highway Systems Application. *Journal of Intelligent Transportation Systems*, **1**, 1-11.

[12] Orda, A. and Rom, R. (1990) Shortest-Path and Minimum-Delay Algorithms in Networks with Time-Dependent Edge-Length. *Journal of the ACM*, **37**, 607-625. http://dx.doi.org/10.1145/79147.214078

[13] Sherali, H., Ozbay, K. and Subramanian, S. (1998) The Time-Dependent Shortest Pair of Disjoint Paths Problem: Complexity, Models, and Algorithms. *Networks*, **31**, 259-272. http://dx.doi.org/10.1002/(SICI)1097-0037(199807)31:4<259::AID-NET6>3.0.CO;2-C

[14] Dean, B. (1999) Continuous-Time Dynamic Shortest Path Algorithms. Master's Thesis, MIT.

[15] Ding, B., Xu, J. and Qin, L. (2008) Finding Time-Dependent Shortest Paths over Large Graphs. *Proceedings of the* 11*th International Conference on Extending Database Technology*, 25-30 March 2008, 205-216.

[16] Kanoulse, E., Du, Y., Xia, T. and Zhang, D. (2006) Finding Fastest Paths on a Road Network with Speed Patterns. *Proceedings of the* 22*nd International Conference on Data Engineering*, Atlanta, 3-7 April 2006, 10-19.

[17] Wagner, D. and Willhalm, T. (2007) Speed-Up Techniques for Shortest-Path Computations. *Lecture Notes in Computer Science*, **4393**, 23-36. http://dx.doi.org/10.1007/978-3-540-70918-3_3