International Conference
20th EURO Mini Conference
“Continuous Optimization and Knowledge-Based Technologies”
(EurOPT-2008)
May 20–23, 2008, Neringa, LITHUANIA

ISBN 978-9955-28-283-9
L. Sakalauskas, G.W. Weber and
E. K. Zavadskas (Eds.): EUROPT-2008
Selected papers. Vilnius, 2008, pp. 48–53

ON SOLVING DYNAMIC SHORTEST PATH PROBLEMS

Ebrahim Nasrabadi, S. Mehdi Hashemi

Department of Computer Sciences, Amirkabir University of Technology,
424 Hafez Ave., Tehran, Iran
E-mail: {nasrabadi,hashemi}@aut.ac.ir

Abstract: Given a dynamic network with n nodes and m arcs in which all attributes including travel times, travel costs and waiting costs may change dynamically over a time horizon T. The dynamic shortest path problem is to determine a path from a specified source node to every other node with minimal total cost, subject to the constraint that the total traverse time is at most T. This problem can be formulated in two ways depending on whether a discrete or continuous representation of time is used. In this paper, we present an $O(nT(n+T))$ time algorithm for solving the discrete-time version of dynamic shortest path problem.

Keywords: Dynamic networks, Dynamic shortest path, Label-setting algorithm

1. Introduction

Dynamic shortest path problem often arise in transportation applications, where travel times and costs change dynamically with time in a predictable fashion. Consider a directed network $G=(N,A)$ with $n=|N|$ nodes and $m=|A|$ arcs. In dynamic problems, a travel time $\tau_{i,j}(t)$ is associated with each arc (i,j) with the following meaning: if one departs from node i at t along the arc (i,j), then $t+\tau_{i,j}(t)$ is the arrival time at node j. In addition to the travel time, a travel cost $c_{i,j}(t)$ may be associated with (i,j) which is the cost of travelling from i to j along the arc (i,j) starting at time t. Let $s \in N$ be a fixed *source* node and t_s be a starting time from that node. The *Dynamic Shortest Path* (DSP) problem is to determine paths of minimal cost from s starting at time t_s to every other node, subject to the constraint that the total traverse time is at most some number T.

DSP were first introduced by Cooke and Halsey (1966) and has been extensively investigated in the literature. Dreyfus (1969) and Kaufman and Smith (1993) studied DSP problem in which all transit times are nondecreasing (known as the First-In-First-Out property) and showed that DSP can be computed by a slightly modified version of any static label-setting or label-correcting shortest path algorithm, where the running time of the modified algorithm is equal to that of its static counterpart. Subsequently, general properties and algorithms have been discussed in both discrete and continuous time-settings by Ahuja *et al.* (2003), Cai, Kloks and Wong (1997), Cai, Sha and Wong (2001), Chabini (1998), Orda and Rom (1991) and Pallottino and Scutella (1998).

Here, we focus on discrete models in which the time variable t varies in the discrete set $\{0,1,\ldots,T\}$. It is a well-known result that waiting at nodes may decrease the cost of paths to the destinations. So a waiting cost $c_i(t)$ may be associated with each node i, which gives the cost of waiting at note i from t to $t+1$. Moreover, we assume that the travel and waiting costs are all nonnegative, whereas transit time can be negative values. In all of the mentioned work except Cai, Sha and Wong (2001), it is assumed that the arc travel times are nonnegative or strictly positive. Here, the authors provided an $O(nmT^2)$ time algorithm for solving DSP problem in which associated travel times and costs may be negative.

DSP problem can be solved by constructing a time-expanded network, which is a static network that encapsulates the time-varying attributes of G. The time-expanded network is formed by creating $T+1$ copies of N, labelled N_0 through N_T, with the t^{th} copy of node i denoted i_t, $t=0,\ldots,T$. In fact, each node in the time-expanded network is actually a node-time pair corresponding to the original node number and time step. For each arc (i,j) in A and each time step t ($0 \le t \le T-\tau_{i,j}(t)$), there is an arc $(i_t, j_t+\tau_{i,j}(t))$ with associated cost $c_{i,j}(t)$. Moreover, for each node i at each time t ($0 \le t \le T-1$), there is a holdover arc (i_t, i_{t+1})

with associated cost $c_i(t)$. Any static path in the constructed time-expanded network corresponds to a path in the original one of equal cost and vice versa. Thus, we may solve DSP problem by solving the classical shortest path problem in the time-expanded network. It is evident that if all travel times are positive, then time-expanded network has no cycle. Thus problem can be solved in $O(mT)$ time. When zero and negative travel times are allowed, some cycle could present in the time-expanded network. Since the costs are nonnegative, one could choose Dijkstra's algorithm (Dijkstra, 1959) on the time-expanded network to solve the DSP problem. The complexity of Dijkstra's algorithm is $O(p^2)$ for a network with p nodes (Ahuja, Magnanti, Orlin, 1993). This complexity becomes $O(n^2T^2)$ when it is applied to the time-expanded network. Although this approach allows us to apply any standard technique on the time-expanded networks, but the size of the time-expanded networks are typically very large for realistic problems and it may be beneficial to avoid such explicit expansion.

In this paper, we present a Dijkstra-like label-setting algorithm with computational complexity time $O(nT(n+T))$ for solving the DSP problem with the nonnegative travel and waiting costs that permits negative and zero travel times. In comparison to time-expanded technique, our approach not only is better from computational point of view, but also it obviates the need for the space-time expansion.

2. The algorithm

We begin this section with two definitions to take into account the existence of time in dynamic networks.

Definition 2.1. Given a directed dynamic network, a dynamic walk is a directed walk $P{:}k=i_1\text{-}i_2\text{-}...,i_r=l$ from k to l together with waiting times $w_{i_1},...,w_{i_r}$ at nodes $i_1,...,i_r$, respectively. Given a starting time t_s, let $\alpha_{i_1}=t_s$, $\beta_{i_1}=\alpha_{i_1}+w_{i_1}$, and define $\alpha_{i_{j+1}}=\beta_{i_j},+\tau_{i_j,i_{j+1}}(\beta_{i_j})$, $\beta_{i_{j+1}}=\alpha_{i_{j+1}}+w_{i_{j+1}}$ for $j=1,...,r-1$. We refer to α_{i_j} as the arrival time and β_{i_j} as the departure time of a node i_j, $j=1,...,r$, on dynamic walk P.

Definition 2.2. Let $P{:}k=i_1,i_2,...,i_r=l$ be a dynamic path from k to l. The *traverse time* and *cost* of P are defined as β_r and $\sum_{j=1}^{r}\sum_{t=\alpha_{i_j}}^{\beta_{i_j-1}}c_{i_j}(t)+\sum_{j=1}^{r-1}c_{i_j,i_{j+1}}(\beta_{i,j})$, respectively. The dynamic path P is said to be feasible if arrival and departure times α_{i_j} and β_{i_j}, $j=1,...,r$, belong to the set $\{0,...,T\}$.

Without ambiguity, in the following we will assume that the length of a path is equal to its cost, and use interchangeably the terminologies cost and length. Moreover, only feasible dynamic paths will be considered in this paper, for the sake of notation simplicity, they will be often referred to simply as dynamic paths.

Since it is assumed that the costs are nonnegative, we propose a time-varying version of the label setting algorithm, first developed by Dijkstra (Dijkstra, 1959) for solving DSP problem. Similar to the Dijkstra's algorithm as described in Ahuja, Magnanti, Orlin (1993), the basic idea in the algorithm is to fan out from s at time t_s and labels nodes at every time step according to their distances from s. A distance label $d_i(t)$ is associated with each node i at each time t. At any point in the algorithm, the distance label $d_i(t)$ is either infinity indicating that we are yet to discover any dynamic path from the source to node i of time t, or it is the length of some dynamic path from the source to node i of time t. The label $d_i(t)$ is permanent once we know that it denotes the length of DSP from s to i of time t; otherwise it is temporary. In addition to label $d_i(t)$, two other labels d_i and β_i are associated with each node i. The label d_i denotes the smallest temporary label corresponding to node i, and label β_i denotes the time of a path to i whose length is d_i. The algorithm associates two pointers $P_i(t)$ and $D_i(t)$ with each node i at time t. $P_i(t)$ denotes the predecessor node of node i along the dynamic path whose length and time are $d_i(t)$ and t, respectively, and $D_i(t)$ denotes the departure time from node $P_i(t)$ corresponding to an arrival time β_i along the path from the source node s to node i of time t. At any iteration, the algorithm selects a node i with minimum temporary label, makes its distance label at time \$\beta_i$, i.e., $d_i(\beta_i)$, permanent, and updates labels and pointers accordingly. Labels $d_1,...,d_n$ allow us to search the minimum temporary label among them instead of the set $\{d_1(t),...,d_n(t):t=0,...,T\}$ with cardinality nT. The algorithm uses a bucket-list $B=\{B_0,...,B_n\backslash\}$ at any iteration in order to efficiently compute d_i and β_i, where B_i contains time steps such *t* that $d_i(t)$ is designed as permanent. The algorithm terminates whenever the minimum temporary label is infinity. At termination, if label $d_i(t)$ is finite then it is optimal, and the corresponding pointers $P_i(t)$ and $D_i(t)$ can be used to reconstruct a DSP starting at time t_s from the source to node *i* of time *t*; otherwise it indicates that residual network contains no dynamic path from *s* to *i* of time *t*. The algorithm formally is presented in the following:

```
algorithm
Step 0. Initialization.
    $B_i:=\{\},\ i\in N,$
    $d_s(t_s):=0,\ P_s(t_s):=0,\ D_s(t_s):=t_s,$
    $d_s(t)=\infty,\ P_s(t):=\infty,\ D_s(t):=\infty,$
    $t\in\{0,...,T\}-\{t_s\},\ d_s:=0,\ \beta_s:=t_s,$
    $d_i(t):=\infty, P_i(t):=\infty, D_i(t):=\infty,\ t\in\{0,...,T\},$
    $d_i:=\infty,\ \beta_i:=\infty,\ \backslash\ i\in N-\{s\},$
Step 1. Node selection, and update labels and pointers.
    let $i$ be a node for which $d_i=\min\{d_j:j\in N\}$,
    if $d_i=\infty$ then go to Step 2,
    $B_i:=B_i\cup\{\beta_i\}$,
    for each $j$ such that $(i,j)\in A$ do
    if $0\le\beta_i+\tau_{i,j}(\beta_i)\le T$ and $d_j(\beta_i+\tau_{i,j}(\beta_i))>d_i(\beta_i)+c_{i,j}(\beta_i)$ then
    begin
            $d_j(\beta_i+\tau_{i,j}(\beta_i)):=d_i(\beta_i)+c_{i,j}(\beta_i)$,
            $P_j(d_i(\beta_i)+\tau_{i,j}(\beta_i)):=i, D_j(d_i(\beta_i)+\tau_{i,j}(\beta_i)):=\beta_i$,
            if $d_j>d_j(d_i(\beta_i)+\tau_{i,j}(\beta_i))$ then
            begin
                    $d_j:=d_j(d_i(\beta_i)+\tau_{i,j}(\beta_i))$,
                    $\beta_i:=\beta_i+\tau_{i,j}(\beta_i)$,
            end
    end
    if $|B_i|\le T$ then
    begin
            if $\beta_i+1\le T$ and $d_i(\beta_i+1)>d_i(\beta_i)+c_i(\beta_i)$ then
            begin
                    $d_i(\beta_i+1):=d_i(\beta_i)+c_i(\beta_i)$,
                    $P_i(\beta_i+1):=i,\ D_i(\beta_i+1):=\beta_i$,
            end
            $\beta_i:=\arg\min\{d_i(t):t\in\{0,...,T\}/B_i\}$,
            $d_i:=d_i(\beta_i)$,
    end
    else
            $d_i:=\infty$,
    Return to Step 1,

Step 2. Reconstructing the determined optimal paths using pointers $P_i(t)$ and $D_i(t)$.
```

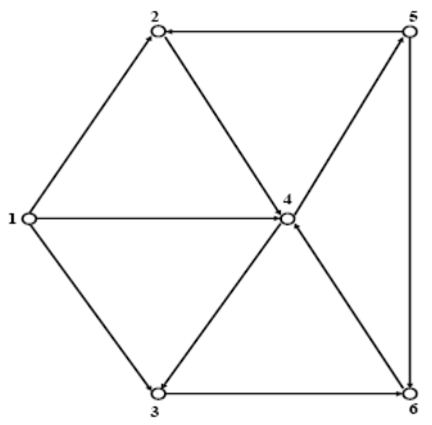

Fig. 1. Network for Example

3. Correctness and complexity of algorithm

In order to prove the correctness of the algorithm, we need to show that at each iteration d_i represents the smallest temporary distance label corresponding to each node i and it is possible to designate the node i at time β_i as permanent whose label d_i is finite and minimum among $\{d1,...,dn\}$. Using an induction argument, we can easily prove the former. Thus, we only focus on the latter. Let i be a node whose label d_i is finite and minimum among $\{d_1,\ldots,d_n\backslash\}$. So there exists a dynamic path such P from s to i of length d_i and time β_i. Since the costs are nonnegative, the length of any dynamic path from s to i of time β_i that contains some nodes as an internal node in which their labels are temporary, will be at least d_i. Thus the label of internal nodes on the DSP from s to i must be designed as permanent. So when the algorithm designs the predecessor node of node i on the DSP, the labels of node i at time β_i is updated accordingly. Consequently, the label β_i is optimal and the path P is the DSP of time β_i from s to i.

The time requirement of the initialization step is bounded by $O(nT)$. The algorithm iterates the node selection step at most nT times and each such iteration requires to find a node with minimum temporary label among a set with cardinality at most n. Thus the total time for node selection is $O(n^2T)$. The required time for the label update step is $|\{j:(i,j)\in A\backslash\}|$ for node i at each time. The algorithm needs to compute the minimum temporary label d_i among a set with cardinality at most T at every iteration. Thus the worst-case complexity of algorithm is $O(nT+n^2T+mT+nT^2)= O(nT(n+T))$.

4. An illustrative example

In this section, an example is illustrated. The network has six nodes and ten arcs connecting those nodes as shown in Figure 1. The travel times, transit times, transit capacities, storage costs and storage capacities are given in Table 1 (a)-(b). Assume that node 1 is the source node and 0 is the starting time. Applying the proposed algorithm, the results are obtained as shown in Table 2 (a)-(c). From these results, one can determine the DSPs from source node 1 starting at time 0 to every other node. For example, the path P:1,4,5,6 with waiting at node 4 from time 2 to 6 is the dynamic shortest path whose traverse cost is 16.

5. Conclusions

In this paper, we have developed a pseudopolynomial time algorithm for solving the DSP problem in dynamic network in which negative and zero travel times are permitted. From the computational and pace-expansion point of view, our approach is superior than raditionally time-expanded network technique.

Table 1 (a). Time-varying travel times and costs for the example

(i, j)	(1,2)	(1,3)	(1,4)	(2,4)	(3,6)	(4,3)	(4,5)	(5,2)	(5,6)	(6,4)
$c_{i,j}(t)$	$5,\ t\le 3,$ $3,\ 4\le t\le 8,$ $4,\ 9\le t\le 11$	$10,\ t\le 6,$ $8,\ 7\le t\le 8,$ $6,\ 9\le t\le 11$	4	$4,\ t\le 3,$ $3,\ 4\le t\le 6,$ $4,\ 7\le t\le 11$	$9,\ t\le 4,$ $7,\ 5\le t\le 8,$ $5,\ 9\le t\le 11$	$6,\ t\le 5,$ $4,\ 6\le t\le 11$	$8,\ t\le 5,$ $2,\ 6\le t\le 11$	2	$5,\ t\le 3,$ $3,\ 4\le t\le 11$	$2,\ t\le 2,$ $4,\ 3\le t\le 11$
$\tau_{i,j}(t)$	$2,\ t\le 5,$ $3,\ 6\le t\le 11$	$1,\ t\le 3,$ $0,\ 4\le t\le 11$	2	$0,\ t\le 4,$ $-2,\ 5\le t\le 1$	3	$2,\ t\le 7,$ $3,\ 8\le t\le 11$	$1,\ t\le 3,$ $0,\ 4\le t\le 11$	$2,\ t\le 3,$ $0,\ 4\le t\le 11$	$-1,\ t\le 4,$ $0,\ 5\le t\le 11$	$2,\ t\le 7,$ $3,\ 8\le t\le 11$

Table 1 (b). Time-varying waiting costs for the example

i	1	2	3	4	5	6
$c_i(t)$	$3,\ t\le 4,$ $2,\ 5\le t\le 11$	3	$1,\ t\le 6,$ $2,\ 7\le t\le 11$	$2,\ t\le 4,$ $1,\ 5\le t\le 11$	2	$2,\ t\le 5,$ $3,\ 6\le t\le 11$

Table 2 (a). Shortest distance labels $d_i(t)$

i \ t	0	1	2	3	4	5	6	7	8	9	10	11
1	0	3	6	9	12	15	17	19	21	23	25	27
2	∞	∞	5	8	11	14	15	16	17	18	19	20
3	∞	10	11	12	10	11	12	13	15	17	19	17
4	∞	∞	4	6	8	10	11	12	13	14	15	16
5	∞	∞	∞	12	14	16	13	14	15	16	17	18
6	∞	∞	17	17	19	19	16	17	18	19	20	21

Table 2 (b). Predecessor nodes $P_i(t)$

i \ t	0	1	2	3	4	5	6	7	8	9	10	11
1	0	1	1	1	1	1	1	1	1	1	1	1
2	∞	∞	1	1	1	1	1	5	5	5	5	5
3	∞	1	3	3	4	3	3	3	3	3	3	4
4	∞	∞	1	4	4	4	4	4	4	4	4	4
5	∞	∞	∞	4	4	5	4	4	4	4	4	4
6	∞	∞	5	5	3	5	5	5	3	3	3	5

Table 2 (c). Predecessor departure times $D_i(t)$

i \ t	0	1	2	3	4	5	6	7	8	9	10	11
1	0	0	1	2	3	4	5	6	7	8	9	10
2	∞	∞	0	1	2	3	4	7	8	9	10	11
3	∞	0	1	2	2	4	5	6	7	8	9	8
4	∞	∞	0	2	3	4	5	6	7	8	9	10
5	∞	∞	∞	2	3	4	6	7	8	9	10	11
6	∞	∞	3	4	1	5	6	7	5	6	7	11

References

Ahuja, R. K.; Magnanti, L. and Orlin, J. B. 1993. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, Englewood Cliffs, New Jersey.

Ahuja, R. K.; Orlin, J. B.; Pallottino, S. and Scutella, M. G. 2003. Dynamic Shortest Paths Minimizing Travel Times and Costs, *Networks* 41: 197–205.

Cai, X.; Kloks, T. and Wong, C. K. 1997. Time-varying shortest path problems with constraints, *Networks*, 29: 141–149.

Cai, X.; Sha, D. and Wong, C. K. 2001. Time-varying minimum cost flow problems, *European Journal of Operational Research* 131: 352–374.

Chabini, L. 1998. Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time, Transportation Research Record 1645: 170–175.

Cooke, L. and Halsey, E. 1966. The shortest route through a network with time-dependent intermodal transit times, *Journal of Mathematical Analysis and Applications* 14: 492–49.

Dijkstra, E. W. 1959. A note on two problems in connection with graphs, *Numerische Mathematik* 1: 269–271.

Dreyfus, S. E. 1090. An appraisal of some shortest-path algorithms, *Operations Research* 17: 395–412.

Kaufman, D. E. and Smith, R. L. 1993. Fastest paths in time-dependent networks for intelligent vehicle-highway systems application, *IVHSJ* 1: 1–11.

Orda, A. and Rom, R. 1990. Shortest-path and minimum-delay algorithms in networks with time-dependent edge length, *Journal of the Association for Computing* 37: 607–625.

Orda, A. and Rom, R. 1991. Minimum weight paths in time-dependent networks, *Networks* 21: 295–320.

Pallottino, S. and Scutella, M. G. 1998. Shortest path algorithms in transportation models: Classical and innovative aspects, in Marcotte, P. and Nguyen, S. (ed.). *Equilibrium and advanced transportation modelling*. Kluwer, Norwell, MA, 245–281.