Using Node Combination Method in Time-expanded Networks

Sahar Abbasi
Department of Industrial Engineering, Najafabad Branch, Islamic Azad University, Esfahan, Iran
Abbasi.iaun.ac@gmail.com

Abstract **- This study concerns the problem of finding shortest paths in time-expanded networks by repeatedly combining the source node's nearest neighbor, time-expanded network is derived from dynamic network G= (V,A,T) and contains one copy of the node set of the underlying 'static' network for each discrete time step (building a time layer). we use node combination (NC) method in networks which arc costs can vary with time, each arc has a transit time and parking with a corresponding time-varying cost is allowed at the nodes. The NC algorithm finds the shortest paths with three simple iterative steps: find the nearest neighbor of the source node, combine that node with the source node, and modify the costs on arcs that connect to the nearest neighbor. The NC algorithm is more comprehensible and convenient for programming as there is no need to maintain a set with the nodes' distances.**

Keywords **-** *shortest path, time-expanded networks, node combination; Node Combination Algorithm*

1. Introduction

The problem of finding the shortest path between two nodes lies at the heart of network flows. It is alluring to both researchers and to practitioners for several reasons: (1) they arise frequently in practice since in a wide variety of application settings we wish to send some material(e.g., a computer data packet, a telephone call, a vehicle) between two specified points in a network as quickly, as cheaply, or as reliably as possible; (2) they are easy to solve efficiently; (3) as the simplest network models, they capture many of the most salient core ingredients of network flows and so they provide both a benchmark and a point of departure for studying more complex network models; and (4) they arise frequently as sub problems when solving many combinatorial and network optimization problems. Even though shortest path problems are relatively easy to solve, the design and analysis of most efficient algorithms for solving them requires considerable ingenuity.

Consequently, the study of shortest path problems is a natural starting point for introducing many key ideas from network flows, including the use of clever data structures and ideas such as data scaling to improve the worst case algorithmic performance [1]. Researchers have studied several different types of (directed) shortest path problems:

1. Finding shortest paths from one node to all other nodes when arc lengths are nonnegative
2. Finding shortest paths from one node to all other nodes for networks with arbitrary arc lengths
3. Finding shortest paths from every node to every other node.
4. Various generalizations of the shortest path problem.

Time-dependent graphs are useful for real word applications. A simple example is that of a computer communications network composed of dial up links each with its individual dialing schedules. Since delays depend on these predetermined schedules, finding the best route for a message from source to destination involves the computation of time-dependent functions [2]. Many types of networks exhibit this kind of dynamic behavior. This paper develops an algorithm to find the dynamic shortest path from the source node to the sink node in acyclic networks with the following specifications. Consider a network that represents a city with the usual rush hour traffic patterns. The dynamic shortest path problem is a generalization of the shortest path problem whose aim is to find a path of minimum cost (length) through a network for which

1. Each arc has a *transit time* which specifies the amount of time to traverse through each arc,
2. Parking (or waiting) is permitted at the nodes of the network for later departure, and Network characteristics such as arc transit times and costs (or length) can change over time and are known for all values of time.

The aim of this paper is to study the dynamic shortest path problem in a discrete time setting with positive transit times. We show that the problem is reduced to a classical shortest path problem on a so-called *time-expanded network*. This allows us to apply algorithms that are available in the classical case to the dynamic case. Then we use Node Combination (NC) algorithm which introduced by Xin Lu in 2011 to implement Dijkstra's algorithm, with which the source node iteratively combines nodes into a new source node and updates the edge weights of the remaining node. When all of the nodes in the connected component of the source node are

finally combined into a single node, the shortest paths from the source node to all other nodes are known. With the method of node combination, the process of finding shortest paths is comparatively simple and much more vivid than with Dijkstra's algorithm [3].

The paper is organized as follows: After review of the shortest path problem in Section 2, we define necessary notation of the dynamic shortest path problem and Node Combination in Section 3, then we use Node Combination (NC) algorithm for solving this problem and summarize our conclusions the related problems in Sect. 4, 5 respectively.

2. Literature Review

Shortest path algorithms have been a subject of extensive research, resulting in a number of algorithms for various conditions and constraints [4–6]. Some algorithms that are based on dynamic programming, zero-one programming and also network flows theory can be found in [7]. Deo and Pang [8] provided a taxonomy and annotation for the shortest path algorithms. When arc lengths are random variables, the problem will become more difficult. Martin found the distribution function of shortest path and also the expected value of shortest path in stochastic networks, in which the arc lengths are independent random variables with polynomial distribution functions, in the form of multiple integrals [9].

Frank computed the probability that the time of the shortest path of the network is smaller than a specific value [10]. He assumed that the arc lengths are continuous random variables. Mirchandani presented another method for obtaining the distribution function of shortest path in stochastic networks [11]. It is not required to solve multiple integrals in this paper, but this method can only be used for the special case where arc lengths are discrete random variables.

Among the various shortest path algorithms developed, Dijkstra's algorithm is probably the most well-known. Though the efficiency and various applications of Dijkstra's algorithm have been widely studied [12], Dijkstra's algorithm may not be easily understood, especially when implementing the labeling method [13].The general properties and algorithms have been discussed in both discrete time and continuous time settings by Ahuja et al. [14], Cai et al. [15], Chabini [16] Orda and Rom [2] among others.

The problem considered in this paper is that of a dynamic network, where the weights (costs) $C_{ij}(t)$ change as a function of time. Given a dynamic network $G = (V, E, T)$ with discrete-time consists of a set of nodes V, ($|V| = n$), node set $V = \{1, 2, \ldots, n\}$, a set of arcs E, ($|E| = m$), *arc set* $E \subseteq V \times V$ and a fixed time horizon $T \in R^{+}$.

We assume that every pair of nodes is connected by at most one arc. Each arc $(i, j) \in E$ has an associated *transit time* $\lambda_{i,j}$, if a vehicle leaves node i at time t along the arc (i, j) then it arrives at node j at time $t + \lambda_{i,j}$. we define a node-time pair to be a member of $V \times \{0,1,\ldots,T-1\}$. A *discrete-time dynamic path* from node-time pair (i, α) to node-time pair (j, β) is a sequence of distinct node-time pairs as $P : (j, \alpha) = (i_1, t_1), (i_2, t_2), \ldots, (i_s , t_s) = (j, \beta)$, in which either $(i_k , i_{k+1}) \in E$ and $t_{k+1} = t_k + \lambda_{ik}$,i_{k+1} , in which case traffic leaves node i_k for node i_{k+1} at time t_k and arrives at t_{k+1}, or $i_k = i_{k+1}$, in which case parking occurs at node i_k at the time step t_{k+1}. Such a sequence is called a *discrete-time dynamic cycle* if $(i, \alpha) = (j, \beta)$ and the other node-time pairs are distinct.

The *cost* of a dynamic path ***P*** is defined by where $c_{i,j}(t)$ is the traversal cost along arc (i, j) at time t, and $f_{ik}(\delta)$ is the parking cost at node i at time t. A path P is said to be a *dynamic shortest path* from to node-time pair (i,α) to node-time pair (j, β), if Cost $[P] \leq$ Cost $[P']$ for all dynamic paths P' from (i,α) to (j, β). We assume that the dynamic network G contains a dynamic path from node-time pair (1,0) to every other node-time pair (i, t) by introducing artificial arcs $(1,i)$ joining node 1 to node i for each node $i \in V \setminus \{1\}$.

Each artificial arc $(1, i)$ has a zero transit time and a large traversal cost. It is clear that no such arc would appear in a dynamic shortest path from (1, 0) to any node-time pair (i,t) unless network G contains no dynamic path from (1,0) to (i,t) without artificial arcs [17].

2.1 Time-expanded Network

Ford and Fulkerson introduce the notion of time-expanded networks. A time-expanded network contains one copy of the node set of the underlying 'static' network for each discrete time step (building a time layer). For a dynamic network $G = (V,A,T)$ the time expanded network $G^T = (V^T,A^T)$ is defined as follows: A *time-expanded network* of G, denoted by $G(\varphi)$,where $\varphi=\{t_0,t_1,\ldots,t_p\}$ contains $p+1$ copies of V, denoted by $V_0, V_1,\ldots, V_p$,in which $Vq\text{-}1$ corresponds to the time step t_{q-1} for $q = 1, \ldots, p-1$, and Vp to the time horizon T.

Subsequently, index q varies from 1 to p. The copy of node $i \in V$ in V_{q-1} is denoted by i_{q-1}. For each arc $(i,j) \in E$ and each time $t_{q-1} \in \varphi$ with $0 \leq t_{q-1}+\lambda_{i,j} \leq T$, Traversing through arc$(i_{q-1}, j_{q'})$ where, $t_{q'} = t_{q-1}+\lambda_{i,j}$ corresponds to leaving node i at time t_{q-1} and arriving at node j at time $t_{q'}$. Hence, arc (i_{q-1}, j_q) has an associated cost $c_{i,j}(t_{q-1})$.

For each node i , there is a holdover arc from i_{q-1} to i_q. Traveling through arc (i_{q-1}, i_q) corresponds to the parking at node i from time t_{q-1} to t_q . So holdover arc (i_{q-1}, i_q) has an associated cost $f_i(t_{q-1})$. An

illustration of a time-expanded network is given in Fig. 1.

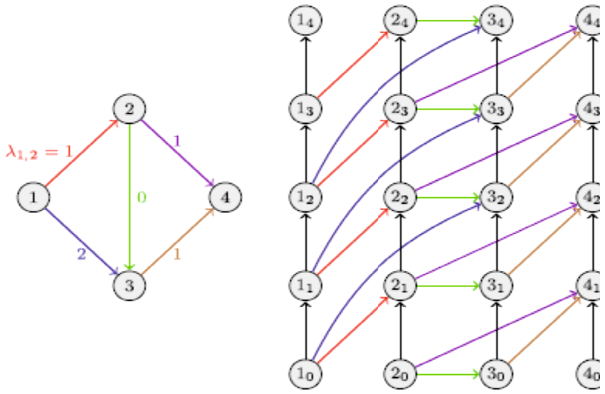

Figure 1. A network G with transit times on the arcs is given on the left hand side.
On the right side, corresponding time-expanded network G (φ) with respect to the partition φ is depicted [18]

2.2 Node Combination

The fundamental idea of the NC algorithm is to combine nodes instead of maintaining the labeling sets in Dijkstra's algorithm. Suppose that all nodes in the network are connected by ropes. The source node is placed in a pool, and other nodes are successively dragged into the pool one by one. Over time, there will be fewer and fewer nodes outside, and finally all nodes will have been dragged into the pool. The combined nodes correspond to the set of solved nodes whose distances have been established in Dijkstra's algorithm. The adjacent neighbors of the combined node correspond to the set of potential nodes from which the closest one is picked. In the meantime, we can update the edge weights to store the distance labels from the source node, instead of maintaining a vector of distances, making the procedure more comprehensible [3].

3. Node Combination Algorithm

Given a nonnegative time-expanded network G^T = (V^T, E^T, C) with *NT* nodes let $C_{NT\times NT}$ be the cost matrix, node-time pair (1,0) be the source node, **d** be the vector whose element $d(i,t_\alpha)$ is to save the distance between source node-time pair (1,0) to node-time pair (i,t_α), then iterations of NC algorithm can be described as follows:

Step 0 **Initialization**. Set d(1,0)=0

Step 1 **Find the nearest neighbor**. Select (1, 1) or (i,t_α) from the neighbors of (1,0), which makes $C_{1i}(0)$= ***min***{ $C(1,0)(1,1)$, $C(1,0)(i,t_\alpha)$}. let $d(i,t_\alpha)$= $C_{1i}(0)$.

If there are no adjacent nodes to (1,0) , stop.

Step 2 **Combine nodes**. Delete (i,t_α) , V=V-(i,t_α).
If $V=\varnothing$, stop.

Step 3 **Modify edge weights**. For each arc-time pair $((i,t_\alpha),(j,t_\beta))$, Update $C(1,0)(i,t_\alpha)$=***min***{ $C(1,0)(i,t_\alpha)$, $C1i(0)+C((i,t_\alpha),(j,t_\beta))$}

Go to Step 1.

Theorem 3.1 NC algorithm solves the Single-Source Shortest Path problem in an increasing order of d (i,t_α).

Theorem 3.2 Given a Time-expanded network G^T = (V^T, E^T, C) with nonnegative arc costs and a source node (1,0) ϵ V^T, NC algorithm computes $d(i,t_\alpha)$ for every (i,t_α) ϵ V^T.
For proofs these theorems refer to [2].

4. Discussions

The NC algorithm can be easily implemented to find the shortest paths, not just the distances.

Let $\boldsymbol{p}_{(1,0)(i,t\alpha)}(1<i<NT)$ be the shortest path from the source node$(1,0)$ to node (i,t_α), $\boldsymbol{u}_{(1,0)(i,t\alpha)}$ be the second last node on $\boldsymbol{p}_{(1,0)(i,t\alpha)}$. To record $\boldsymbol{u}_{(1,0)(i,t\alpha)}$ we can declare a vector $\boldsymbol{P}$ with length of NT , and initialize all the elements as(1,0). If $\boldsymbol{C}_{(1,0)(i,t\alpha)}$ is updated in Step 3 **(** $\boldsymbol{C}_{(1,0)(i,t\alpha)} \leftarrow \boldsymbol{C}_{(1,0)(j,t\beta)}\, \boldsymbol{C}_{(j,t\beta)(k,t\theta)}$**)**, set $\boldsymbol{P}_{(j,t\beta)}=(k,t_\theta)$.

When the NC algorithm terminates, *P* records the information of shortest paths between and all the other nodes. To find the shortest path between node-time pair (1,0) and node-time pair (i,t_α). We can trace from $\boldsymbol{P}_{(j,t\beta)}$: if $\boldsymbol{u}_{(1,0)(j,t\beta)}=(k,t_\theta)$, than $\boldsymbol{u}_{(1,0)(k,t\theta)}=\boldsymbol{P}_{(k,t\theta)}$,…, till
$\boldsymbol{P}_{(k,t\theta)}$=(1,0).

The shortest path is:
$(1,0),\ldots, P(P(P(j,t\beta))),\ldots, P(P(j,t\beta)),P(j,t\beta), (j,t\beta)$

5. Conclusions

In this paper we considered the dynamic shortest path problem, motivated by its applications in dynamic minimum cost flows. We showed that this problem is equivalent to a classical shortest path problem in a so-called time-expanded network. Using the NC algorithm, we found the shortest path by node combination instead of by labeling operations. The difference between the NC algorithm and Dijkstra's algorithm is, first, the set of visited (solved) nodes whose distances have been established. In the NC algorithm, nodes are combined into the new source node, which means that we need not maintain this set.

Second, the relaxation is done on the arc cost directly, which means that no additional memory or CPU-cycles are needed to record the temporary distances. Third, the NC algorithm is carried out by repeatedly finding the source node's nearest neighbor, which makes the process of finding shortest paths more comprehensible and vivid. Node combination makes the process of finding the shortest paths much more straightforward, Comprehensible, and memory-sparing.

References

[1] Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice-Hall, Inc., New Jersey (1993).

[2] A.Orda, R. Rom, *Distributed shortest-path protocols for time-dependent networks*, Distributed Computing 10 (1) (1996) 49–62.

[3] Xin Lu, Martin Camitz, *Finding the Shortest Paths by Node Combination*, Appl. Math. Comput. (2011), doi:10.1016/j.amc.2011.01.019.

[4] E.W. Dijkstra, *A note on two papers in connection with graphs*, Numeriske Mathematics 1 (1959) 269– 271.

[5] D. Eppstein, *Finding the k shortest paths*, SIAM Journal on Computing 28 (2) (1998) 653–674.

[6] R.W. Floyd, Algorithm 97: Shortest paths, Communications of the ACM 5 (1962) 345.

[7] M. Bazaraa, J. Jarvis, H. Sherali, *Linear Programming and Network Flows*, second ed., Wiley, New York, 1990.

[8] J. Martin, *Distribution of time through a directed acyclic network*, Operations Research 13 (1965) 46– 66.

[9] N. Deo, C. Pang, *Shortest path algorithms*: Taxonomy and annotation, Networks 14 (1984) 275–323.

[10] H. Frank, *Shortest paths in probabilistic graphs*, Operations Research 17 (1969) 583–599.

[11] P. Mirchandani, *Shortest distance and reliability of probabilistic networks*, Computer and Operations Research 3 (1976) 347–355.

[12] B.V. Cherkassky, A.V. Goldberg and T. Radzik, *Shortest paths algorithms: Theory and experimental evaluation.Mathematical Programming*, 1996. 73(2): 129-174.

[13] F.B. Zhan, *Three fastest shortest path algorithms on real road networks: Data structures and procedures*. Journal of Geographic Information and Decision Analysis, 2001. 1(1): 69-82.

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

[15] Cai, X., Kloks, T., Wong, C.K.*: Time-varying shortest path problems with constraints*. Networks29, 141–149 (1997).

[16] Chabini, L.: *Discrete dynamic shortest path problems in transportation applications*: Complexity and algorithms with optimal run time. Transp. Res. Rec. 1645, 170–175 (1998).

[17] Abbasi. S ,Ibrahimnejad. S, *Finding the Shortest Path in Dynamic Network using Labeling Algorithm*, International Journal of Business and Social Science, Vol. 2 No. 20; November 2011.

[18] S. Mehdi Hashemi, Shaghayegh Mokarami, Ebrahim Nasrabadi, *Dynamic shortest path problems with time-varying costs*. Optim Lett 4,147–156(2010).