A DYNAMIC NAVIGATION ALGORITHM CONSIDERING NETWORK DISRUPTIONS

Jincheng Jiang[a] and Lixin Wu[b*]

[a] Academy of Disaster Reduction and Emergency Management, Beijing Normal University, Beijing 100875, China

[b] IoT Perception Mine Research Center, China University of Mining and Technology, Xuzhou, 221008, China

KEY WORDS: shortest path, dynamic navigation, emergency evacuation, stochastic disruption, priori knowledge

ABSTRACT:

In traffic network, link disruptions or recoveries caused by sudden accidents, bad weather and traffic congestion, lead to significant increase or decrease in travel times on some network links. Similar situation also occurs in real-time emergency evacuation plan in indoor areas. As the dynamic nature of real-time network information generates better navigation solutions than the static one, a real-time dynamic navigation algorithm for emergency evacuation with stochastic disruptions or recoveries in the network is presented in this paper. Compared with traditional existing algorithms, this new algorithm adjusts pre-existing path to a new optimal one according to the changing link travel time. With real-time network information, it can provide the optional path quickly to adapt to the rapid changing network properties. Theoretical analysis and experimental results demonstrate that this proposed algorithm performs a high time efficiency to get exact solution and indirect information can be calculated in spare time.

1 Introduction

Emergency evacuation, including traffic evacuation (Hamza-Lup, 2007), earthquake evacuation (Hao-Che Wu, 2012) and 3D building evacuation (Chalmet, 1982) etc., are common in our daily lives. In emergency situations, the unforeseen events such as vehicle breakdowns, traffic accidents, fire hazards, roadway conditions etc., contribute to the uncertainties of travel times in real-life transportation networks (Sahar Abbasi, 2011). Thus, the dynamic nature of real-time network information generates better navigation solutions than the static one. For an individual traveler, his/her interest is to find a real-time optimal path with shortest distance, minimal time or minimum fuel consumption, etc. Fortunately, the shortest path algorithm is the solution to this issue, whose goal is to find the optimal path between two nodes in a given network while minimizing the total cost (travel times).

A large body of literatures on the shortest path algorithm for static and dynamic networks (Powell W.B., 1996) has been presented. Actually, most dynamic shortest path algorithms are the variants or improvements of static algorithms. In general, they can be divided into two categories (Cho H., 2009): exact and heuristic algorithm. For the exact algorithm, Dijkstra algorithm (Dijkstra E. W., 1959) remains by far one of the fastest known algorithms for the general case of arbitrary nonnegative link cost network. Dijkstra algorithm starts from *origin* node and gradually expending the search space to neighboring links until the *destination* node with the breadth-first search method. The time complexities of Dijkstra algorithm are $O(n^2+m)$, $O((n+m)\log n)$ and $O(m+n\log n)$ respectively when the data structures are linear array, binary and Fibonacci heap. Another famous exact algorithm is Bellman-Ford algorithm (Bellman R., 1958), which can deal with the links with negative cost. Bellman-Ford algorithm begins from any node and finds the shortest path between the *origin* and every node located within *k* links after *k*th iterations utilizing a link relaxation method. Its time complexity is $O(nm)$, which is worse than the Dijkstra algorithm. Some other common exact shortest path algorithms include Dantzig (G. B. Dantzig, 1967), D'Esopo-Pape (S. Pallottino, 1979), Pallottino (G. GaUo and S. Pallottino, 1986). More details were introduced by Lu feng (2001). However, the exact algorithms examine all possible nodes from the *origin* to the *destination* without direction guide, resulting in many unnecessary operations, which makes them cannot meet the instantaneity demand of real-time application.

In emergency situations, the shortest paths need to be quickly identified because an immediate response is required. To overcome the defect of exact algorithm, lots of heuristic methods have been proposed to decrease the computation time of the shortest path algorithm. The advantage of heuristic algorithms lie in that they utilize prior information contained in the network structure, e.g. the location of *origin* node and *destination* node (L. Fu, 2006). A*

Jincheng Jiang,
E-mail address: jiangjincheng0305@126.com
Lixin Wu,
E-mail address: awulixin@263.net

algorithm (Hart P. E., 1968) is one of the most effective methods of utilizing the idea of limiting search space. By calculating an estimated cost based on the Euclidian distance between the *origin* and *destination* nodes, the search direction of the A* algorithm is guided to the destination. Not only the search space has been reduced compared with exact algorithm, but also the A* algorithm can obtain a relatively high accuracy. Thus, A* algorithm has been widely implemented in vehicle navigation system. Another heuristic method of limiting search space is hierarchical search algorithm (Huang Y, 1996) that divides the network into several sub-networks with different scales. It first calculates a rough shortest path with a relatively low accuracy and more accurate calculations are completed later with more details in fine scale. All aforementioned heuristic algorithms adopt the limiting search space strategy to reduce the computational complexity. While the bi-directional search algorithm (Dantzig G. B., 1960) is to decompose the search problem into two separate produces. Its basic idea is expending the search space from both *origin* and *destination* nodes concurrently. The computational complexity and accuracy depend on the number of intersected nodes between two search sub-spaces. Although heuristic methods achieve substantial performance improvements, they come at the cost of accuracy.

Both the exact and heuristics algorithms have defects: the exact shortest path algorithms are too computationally intensive to be feasible for real-time operations; the heuristics algorithms could reduce the search space or decompose the search problem by utilizing location information contained in the network structure, but they always cannot get the optimal solution, thus the travel time provided for users may be longer due to accuracy loss compared with exact algorithm. The computational time and travel time are both important for traveler. Neither of exact and heuristics algorithms can maximize two parts of time. In emergency situations, the traffic flow is real-time dynamics, and the navigation path changes from one path to another one. In this paper, we focus on the exact dynamic adjusting method (EDAM) for the shortest navigation path considering the network disruptions (Derya Sever, 2013) and recoveries. The EDAM is inspired by the network simplex method (Damian J. Kelly, 1993; Helgason R. 1995). With the EDAM, the shortest path is calculated dynamically with variation flow, making the best of priori knowledge including the location of origin and destination, network structure and pre-existing path.

This paper is organized as follow. In next section, the shortest path problem with stochastic link disruptions and recoveries is introduced, and then the EDAM is presented, also theoretical analysis of time complexity and accuracy are given. In Section 3, experiments are performed, followed by concluding remarks.

2 Methodology

2.1 The shortest path problem

Let $G(N,A)$ be a network with $n=|N|$ nodes and $m=|A|$ links, where N is a set of node and A is the link set. Each link $(i,j)\in A$ has a cost c_{ij}, which represents the travel time for a traveler to go through that link. A path *P* from *origin* (*O*) to *destination* (*D*) is a series of connected link and the travel time of *P* is the sum of travel time on every individual link in *P*. The shortest path problem is to find the path with minimum total travel time. Let d_i denotes the shortest distance (minimum total travel time) from node *i* to *D*. Then, the following constraints for each node *i* should be met:

$$d_i \le d_j + c_{ij}, \qquad (i,j)\in A \tag{1}$$

$$d_D = 0 \tag{2}$$

In a dynamic network, the travel time varies over time (X. Cai, 1997). Generally, it is assumed that the link travel times on all links are random variables, and a network is referred to dynamic and stochastic network (L. Fu, 1996) if the link travel time is modeled as a stochastic process. In this paper, we focus on the case where the network flow is approximate equilibrium within a short time, and only a few of link travel time change concurrently during this period. Considering the link disruptions and recoveries, the travel times could both increase or decrease.

For the expression convenience, some definitions and notations are given here. Let T_D denote the shortest path tree (solid lines in Fig. 1) (P. Narva ́ez, 2000). In this tree, P_i is the precursor node in T_D of node *i*, C_i is the set of successor nodes in T_D of node *i*, reduced cost $c_{uv}^d = d_v + c_{uv} - d_u$. In Fig. 1, $P_1 = 4$, $C_1 = \{O, 2\}$, $d_1 = 12$, $d_0 = 5$, $c_{01}^d = d_1 + c_{01} - d_0 = 8$. A sub-tree T_i contains all nodes whose shortest paths to *D* pass through *i*, e.g. T_7 contain node *O*, 1, 2 and 4.

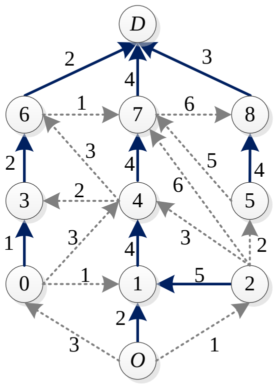

Figure 1. The shortest path tree

2.2 Dynamic Navigation Algorithm

When an emergency situation occurs, two parts of work need to be done for the dynamic navigation algorithm. At the beginning, only network topology and instant travel times on all links without any predictive information are acquired. With relate knowledge, an exact shortest path tree will be constructed. And then based on foregone shortest path tree, the shortest path from *O* to *D* could be updated according to the real-time network information.

(1) Generating the initial solution

As the introduction in section 1, neither of exact and heuristic algorithms can maximize both the computational time and travel time. However, exact solution (i.e. exact shortest path tree) need to be obtained in this step because the accuracy is important for subsequent calculations. Compare with total evacuation time, the computation time is much less. In addition, the travel time directly affects the travel cost. So the accuracy of the shortest path algorithm is relatively more important than the time performance.

To compromise the contradiction of time performance and accuracy, two strategies are often taken. One is adopting hybrid shortest path algorithm (Hsun-Jung Cho, 2009) that integrates several exact and heuristics algorithms. Two advantages exist: 1) although the hybrid shortest path is not always the optimal solution, it is very close to optimal; 2) a significant performance improvement compared with single exact algorithm is made. The other strategy is calculating the shortest paths with exact and heuristics algorithms concurrently by utilizing parallel computing technology. The heuristics shortest path could be used for navigation at first, and replaced by the exact one once the exact algorithm is completed.

(2) Real-time updating method

After the first step, the shortest path tree T_D has been constructed. With the latest and real-time network information, the shortest path could be updated dynamically for real-time navigation. Considering the network link disruptions or recoveries, the correspondent link travel times can both increase and decrease. If the network link is disrupted, the travel time is set to infinity.

When the travel time of link $(i,j) \in A$ changes over time regardless of increases or decreases, the recalculation is triggered. The main steps of the EDAM are as follow:

```
EDAM
  1) let $\Delta = d_i - (c_{ij} + d_j)$;
  2) if $\Delta > 0$, then set $d_i = d_i + \Delta$, $P_i = j$
     and $d_k = d_k + \Delta$ for any $k \in T_i$;
  3) if $\Delta < 0$ and $(i,j) \in T_D$, then
     a) let $T' = T_i$;
     b) find the minimum reduced cost $c_{uv}^d$
        where $u \in T'$and $v \notin T'$;
     c) if $c_{uv}^d < \Delta$, the algorithm terminates;
        otherwise, set $d_u = d_u + c_{uv}^d$,
        $P_u = v$, $T' = T' - T_u$;
     d) perform step b and c repeatedly until
        $T' = \emptyset$.
```

In the above steps, if i is the ancestor of j, then (j,i) is regarded as the changed link, and then (i,j) case is disposed. When a travel time changes, corresponding recalculations are triggered. The most time-consuming case is $\Delta < 0$ and $(i,j) \in T_D$. Under this circumstance, the $O - D$ shortest path can be outputted to user once $O \notin T'$ and the rest of calculation can be done in background. With this optimizing, user always obtains the optimal navigation route in the earliest time.

2.3 Analysis

The EDAM can obtain the optimal solution for First-in First-out network with varying cost. In a shortest path tree T_D, four main cases exist when the travel time changes: c_{ij} increases or decreases for $(i,j) \in T_D$ or $(i,j) \notin T_D$ respectively. When $(i,j) \notin T_D$, if $d_i \leq d_j + c_{ij}$, no shortest paths will be influenced; while if $d_i > d_j + c_{ij}$, the precursor of node i will turn to j and all shortest distances for any node $k \in T_i$ should be reduced by $d_j + c_{ij} - d_i$. When $(i,j) \in T_D$, if c_{ij} decreases, all shortest paths keep constant, but the shortest distances for all nodes in T_i should be updated by the variation of c_{ij}; if c_{ij} increases, then all shortest paths for node $k \in T_i$ may change, and the steps shown in the EDAM can guarantee the correctness of shortest path tree. For example, if c_{7D} increases to 6 and c_{36}^d is the maximum reduced cost, then $P(3,6,D)$ is the shortest path of node 3 because all shortest paths of node $k_1 \notin T_7$ keep constant and all reduced cost c_{uv}^d where $u \in T_7$ and $v \in T_7$ is zero. Thus, each renewal of shortest path tree can obtain optimal solution.

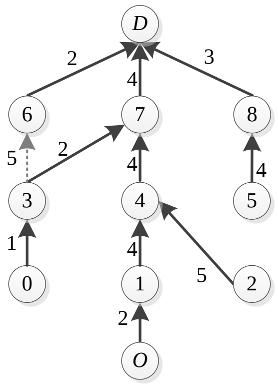

Figure 2. The update method of the shortest path tree

As for the time complexity, the above computational steps show that only when the travel time in the shortest path increases to a certain level, more than one iteration are needed to be performed. In this case, the computational complexity for each node $k \in T_i$ depends on the topological length of shortest path to i; otherwise, the shortest path remain unchanged. In the bad case, two main steps are included. At first, searching the maximum reduced cost c_{uv}^d where $u \in T'$ and $v \notin T'$ with $|T_i|$ time complexity is performed. Then, T' is converted into $\emptyset$ through replaced by $T' = T' - T_u$ iteratively, additional $|T_u|$ space need to be searched at each iteration. The total time complexity of updating shortest path is $2|T_i|$. In the above steps, the shortest path and distance are recalculated for all nodes in T_i. As descripted in section 3, the shortest path from O to D could be outputted to user at the earliest time. If $\Delta > 0$, all shortest paths keep constant, thus the user is navigated along original route. While if $\Delta < 0$ and $(i,j) \in T_D$, the time complexity of finding the maximum reduced cost c_{uv}^d and adjusting the shortest path is $O(2|T_i|)$. To maintain the shortest path tree, the shortest path and distance for every node

$k \in T_i$ should be updated. Once $k \in T_i - T_u$, the shortest path of O is obtained, and the remaining works could be done in the background.

3 Experiment

3.1 Experimental design

The test data sets in our experiment are artificial networks, including a *Washington-Line-Moderate* (WLM) network with 4098 nodes and 65,028 links, a *Washington-RLG-Long* (WRL) network with 8194 nodes and 24,512 links, a *Washington-RLG-Wide* (WRW) network with 8,194 nodes and 24,448 links, a *Genrmf-Long* (GL) network with 4,096 nodes and 18,368 links, a *Genrmf-Wide* (GW) network with 8,214 nodes and 38,813 links, and an *Acyclic-Dense* (AD) network with 256 nodes and 32,640 links, *AK* network with 4,102 nodes and 6,151 links. All of them were provided by *the first DIMACS implementation challenge* and detailed introductions are given by Boris V Cherkassky (1995).

To verify the correctness and measure the time performance of the EDAM, the experiment is designed as follow: 1) every shortest path tree updated by the EDAM will be compared with the result of the traditional Dijkstra algorithm; 2) the consumed time of adjusting shortest path will be tested considering the travel time increases and decreases in random selected links. In the experiment, after obtaining the initial shortest path tree with the Dijkstra algorithm, 100,000 number of change are generated randomly, and the EDAM adjusts the shortest paths according to the changes. Correspondingly, the Dijkstra algorithm recalculates all works on the basis of same changes.

All algorithms were executed in the same computing environment, which is a personal computer with 2.50GHz CPU, 10GB memory and 64-bit Windows 7 Professional operating system. All codes are written in C++ and compiled with the *gcc* compiler.

3.2 Results and discussion

Both the correctness and time performance have been tested. For the correctness, the distances from the *origin* to *destination* nodes calculated by the EDAM and Dijkstra algorithm are the same. Thus, the shortest paths of the EDAM are exact, not approximate.

	WLM	WRL	WRW	GL	GW	AD	AK
Dijkstra (ms)	1,403,940	576,748	437,787	385,176	363,628	85,426	34,080
EDAM (ms)	10,069	16,564	9,604	8,740	9,964	309	9,674
Case 1	166,780	116,735	117,039	125,953	120,564	196,341	43,772
Case 2	15,278	14,527	14,966	14,700	18,920	1,649	7,851
Case 3	5,229	21,392	21,428	20,322	20,034	694	35,835
Case 4	1,013	9,450	9,732	8,241	8,620	56	25,495
Others	11,700	37,896	36,835	30,784	31,862	1,260	87,047

Table 1. The time performance of the EDAM

Table 1 shows the time performance of the EDAM and Dijkstra algorithm, it demonstrates that the consumed time of each adjusting operation in the EDAM is less than 0.1 milliseconds averagely. In table 1, case 1 denotes $(i,j) \notin T_D$ and c_{ij} increases; case 2 denotes $(i,j) \notin T_D$ and c_{ij} decreases; case 3 denotes $(i,j) \in T_D$ and c_{ij} increases; case 4 denotes $(i,j) \in T_D$ and c_{ij} decreases; while other cases include the variation is zero or *i* is an ancestor node of *j*. The frequency of each case depends on the network density, for example, the frequencies of case 3 and 4 are relative low in AD network, and high in AK network because AD is dense network and AK is sparse network. The total number of all cases is twice as total changes, because each change will trigger two cases for (i,j) and (j,i).

When compared with the Dijkstra algorithm, the EDAM runs much faster although unfair exists. The EDAM can deal with the cases where the link travel time can either increase or decrease, but the Dijkstra algorithm can make the best of priori knowledge only when the link travel time decreases.

4 Conclusions and future work

This paper presents a novel navigation algorithm for emergency evacuation, which is adapted to the dynamic changed network weights. This algorithm minimizes both of the computing time and evacuation time. Making the best of pre-existing shortest path tree, the change of travel time is classified into four categories and different cases are recalculated with different strategies. Furthermore, the shortest *O-D* path is always outputted to traveler at the earliest time. Theoretical analysis show that the optimal solution can be obtained and the total time complexity of updating shortest path tree is just $2|T_i|$ in the worst case, where *i* is the node related to changed link. The experimental result demonstrated that the EDAM runs much faster than the traditional Dijkstra algorithm with obtaining the optimal solution. With the help of global positioning system and geography information system, the EDAM can provide a real-time navigation function for traveler.

However, this paper just processes the case where only one link travel time changes at one time. In the reality, more than one link travel time are about to change concurrently. In future, we focus on the efficient algorithm of dealing with multi-varying cases. In addition, the predictive information has not been taken into consideration (Ahuja, 2003). Therefore, effective algorithms will be designed for dynamic stochastic network with predicted traffic condition.

Acknowledgements This study was funded by the National High-Tech Research and Development Program of China (2011AA120302).

5 Reference

Hamza-Lup, G.L., Hua, K.A., Peng, R., 2007, Leveraging e-transportation in real-time traffic evacuation management. *Electronic Commerce Research and Applications*, 6 (4), 413 – 424.

Hao-Che Wu, Michael K., Lindell, Carla S. Prater, 2012, Logistics of hurricane evacuation in Hurricanes Katrina and Rita, *Transportation Research Part F*, 15: 445-461.

Chalmet, L. G., R. L. Francis, P. B. Saunders, 1982, Network models for building evacuation. *Management Sci*, 28: 86 –105.

Sahar Abbasi, Sadoullah Ebrahimnejad, 2011, Finding the Shortest Path in Dynamic Network using Labeling Algorithm, *International Journal of Business and Social Science*, 2(20): 239-243.

Powell W.B., Cheung R.K., 1996, An algorithm for multistage dynamic networks with random arc capacities, with an application to dynamic fleet management. *Opns Res*, 44: 951 – 963.

Cho H. J., Lan C. L., 2009, Hybrid shortest path algorithm for vehicle navigation. *J Supercomput*, 49: 234–247.

Dijkstra E. W., 1959, A note on two problems in connexion with graphs. *Numer Math*, 1: 269–271.

Bellman R., 1958, On a routing problem. *Quarterly of Applied Mathematics*, 16(1): 87-90.

G. B. Dantzig, 1966, All shortest routes in a graph, Theory of Graphs, *Int. Sym. Rome,* 91-92.

S. Pallottino, 1979, Adaptation de l' Algorithme de d'Esopo-Pape pour ta determination de tousles chemins les plus courts: ameliorations et simplifications, CRT, University of Montreal, Publ. No. 136.

G. GaUo, S. Pallottino, 1986, Shortest path methods: A unifying approach, *Mathematical Programming Study,* 26: 38-64.

Lu Feng, 2001, Shortest Path Algorithms: Taxonomy and Advance in Research, *Acta Geodaetica et Cartographica Sinica*, 30(3): 269-275.

L. Fu, D. Sun, L.R. Rilett, 2006, Heuristic shortest path algorithms for transportation applications: State of the art, *Computers & Operations Research*, 33: 3324-3343.

Hart P. E., et al., 1968, A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics SSC4*, 4(2): 100-107.

Huang Y, Jing N, 1996, Evaluation of hierarchical path finding techniques for ITS route guidance. *In: Proceeding of annual meeting of ITS America,* 340–350.

Dantzig G. B., 1960, On the shortest route through a network, *Manag Sci*, 6: 87–90.

Derya Sever, Nico Dellaert, Tomvan Woensel, Ton de Kok, Dynamic shortest path problems: Hybrid routing policies considering network disruptions, *EJOR*, 2013, 40 (12): 2852-2863.

Damian J. Kelly, Garrett M. O'Neill, 1993, The minimum Cost Flow Problem and The Network Simplex Solution Method, Master Degree Dissertation, University College, Dublin.

Helgason R., Kennington J., 1995, Primal Simplex Algorithms for Minimum Cost Network Flows. *Handbook on Operations Research and Management Science*, Amsterdam, 7: 85-133.

X. Cai, T. Kloks, C.K. Wong, 1997, Time-varying shortest path problems with constraints, *Networks*, 29 (3): 141–149.

L. Fu, 1996, Real-time vehicle routing and scheduling in dynamic and stochastic networks, Ph.D. Thesis at the University of Alberta.

P. Narva ́ez, K. Y. Siu, H. Y. Tzeng, 2000, New Dynamic Algorithms for Shortest Path Tree Computation, *IEEE/ACM Trans. Networking,* 8(6): 734-746.

Hsun-Jung Cho, Chien-Lun Lan, 2009, Hybrid shortest path algorithm for vehicle navigation, *The Journal of Supercomputing*, 49: 234-247.

Boris V Cherkassky, Andrew V Goldberg, 1995, On implementing the push-relabel method for the maximum flow problem, *Lecture Notes in Computer Science*, 920: 157-171

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