Dynamic Routing during Disaster Events

Siska Fitrianie  
Delft University of Technology  
s.fitrianie@tudelft.nl

Leon J. M. Rothkrantz  
Delft University of Technology  
Czech Technical University in Prague  
l.j.m.rothkrantz@tudelft.nl

ABSTRACT
Innovations in mobile technology allow the use of Internet and smartphones for communicating disasters and coordinating evacuations. However, given the turbulent nature of disaster situations, the people and systems at crisis center are subjected to information overload, which can obstruct timely and accurate information sharing. A dynamic and automated evacuation plan that is able to predict future disaster outcome can be used to coordinate the affected people to safety in times of crisis. In this paper, we present a dynamic version of the shortest path algorithm of Dijkstra. The algorithm is able to compute the shortest path from the user’s location (sent by the smartphone) to the safety area by taking into account possible affected areas in future. We aim at employing the computed routes on our mobile communication system for navigating affected people during emergency and disaster evacuations. Two simulation studies have validated the performance of the developed algorithm.

Keywords
Evacuation, disaster events, dynamic routing, mitigation planning

INTRODUCTION
Major disasters can cause widespread destruction and destroys human lives and properties. A disaster evacuation may be carried out before, during or after such events. The effort can involve a huge number of people. However, the disasters can significantly change an area dynamically, render earlier geographical data obsolete, and make damaged infrastructure hazardous to people. In order to minimize public exposure to such dangerous conditions, which prevail in disaster area, people need to be guided to safer areas. Thus, a timely and context-based navigation route that provides sufficient and flexible guidance given the dynamic altered environment in disaster area is critical in this situation.

The current practice of disaster evacuation relies on human operators in crisis center for coordination and relaying information about situations in disaster areas. Based on the field observation focusing on the process of creating disaster situation maps, Gunawan et al. (2009) reported that the hierarchical structure of the current disaster management model has delayed and complicated the knowledge sharing and decision-making. Moreover, the extent of the damages and the limited and overwhelmed emergency services can cause rendering the situation awareness both out-of-date and inaccurate.

Recent events have shown that people in an emergency or a disaster were able to creatively utilize familiar technology, or quickly adopt new unfamiliar ones for their purpose (Hughes & Palen, 2009). The use of Internet and mobile phones allowed community collaboration through crowdsourcing and journalism activities brought together to communicate disasters via popular media (Palen et al., 2009). Experiences from previous major disaster events and research experimental results showed that such a distributed approach that utilizes citizen
participation for collecting situation data reports in the field was more effective, efficient and preferred (Gunawan et al., 2012). The resulted observational data can be used for creating dynamic situational maps given the altered environment in the disaster area.

In this paper, we discuss about a dynamic version of the well-known Dijkstra algorithm to compute the shortest route to safe areas. Instead of computing the shortest route at regular times, here, we have added the time axis as the third dimension to compute the shortest route in the future by taking into account possible changes altered the environment. We aim at using the computed shortest route information on a dynamic routing system offered via smartphones. The input of the system is actual observational reports from people while walking along the route (i.e. what the observers see, hear, smell, and even feel or experience, e.g. blocked road, broken bridge, smoke, and so on). The route will be dynamically computed based on the reports. The current version of the algorithm includes the plume model that can be used to predict the contaminated areas in time and to compute the shortest routes to safe areas as well as to provide update information to people in the affected area.

The outline of this paper is as follows. In the next section we describe related previous work: the agent-based communication network that allows context-mediated collaboration between mobile users, the static and dynamic routing algorithms that were applied in the field of transportation networks, the iconic crisis language that is used for reporting observation along the path to the safety area, and the gas dispersion model that is used to predict the path of disaster. Then, we present the dynamic routing algorithm and two experiments to validate the algorithm.

PREVIOUS WORK
Agent-based Communication Network for Collaboration

Rothkrantz et al. 2005 developed a hybrid communication network architecture based on multi- and mobile agents. The architecture supported context-mediated social awareness between mobile systems. It took the assumption that each individual in the field is equipped with a mobile device (e.g. smartphones) that could communicate with other mobile devices. A personal agent would supervise each user. To cope with nondeterministic environments during disaster resulting from the global wired-communication breakdown, mobile ad-hoc network was applied. The network enabled users to connect to the server and communicate among themselves by using ad hoc wireless networks. In Rothkrantz et al. (2014), the architecture was expanded to allow different emergent technologies developed on smartphones. Two of these technologies are an iconic crisis language for reporting technology and a personalized dynamic routing system.

Static and Dynamic Routing

In Rothkrantz (2014), routing algorithms were applied in the field of network transportation. A static routing algorithm can be used to route car drivers in a static environment from A to B along the shortest path. In the case of incidents on the road where traffic jams and congestion occur, the shortest route may not be always the fastest route. A dynamic routing algorithm based on ideas from Artificial Life, called Ant Based Control (ABC), was proposed. The algorithm is modeled based on the trail-laying abilities of natural ants. The goal of the algorithm is to route the traffic as soon as possible out of a certain zone via some exit points. We found that the dynamic changing environments of the ABC algorithm are rather slow. It needs a huge number of users to adapt to changing environments. Another problem was that the input of the ABC would be only based on the current state of the art and would not take into account to be expected changes of the network architecture.

The dynamic Dijkstra algorithm used in this paper was based on the work of Tatamir et al. (2009) and Radu et al. (2013). The previous work employed the algorithm to route pedestrians via their smartphones. They utilized information about dynamic changing traffic speed on the roads. This information is computed by tracking car drivers using routing devices on their smartphones. To predict traffic delays in the future, they used probabilistic prediction models and historic traffic database. Contrast to the previous work, the innovative part of the present paper is that future changes in the road network is computed based on observation reports from affected people using their smartphones and models of hazardous substance dispersion, i.e. toxic gas dispersion.
Iconic Crisis Language

Fitrianie & Rothkrantz (2005, 2007) and Fitrianie et al. (2006, 2008) have developed some versions of an icon interface and a dedicated icon-based language for reporting observation on smartphones. Fitrianie (2010) showed that the interface was able to support communication between users. The human users were able to express their concepts in mind solely using a spatial arrangement of icons. Communicating observational reports with icons was chosen because icons offer a potential across language barriers and a direct method for conversion with other modalities. As pictorial signs, icons can be recognized quickly; therefore, they can evoke a readiness to respond and quick ensuing actions.

The interface supports a real-time two-way context-sensitive data exchange. It enables users to report their observations from their position. Each observation sent from the user’s smartphones to the crisis center contains its timestamp and GPS-based location. Observational data, such as blocked road, broken bridge, damaged building, fire, and so on, can be used by the dynamic routing algorithm to dynamically calculate safe routes. At the same time, the icon-based interface receives the new computed route, which were calculated based on information gathered from all other users. This two-way communication allows both operators in crisis center and people in the field to continuously update of the state of the art of the situations. Figure 1 shows the map-based icon interface and some examples of icons.

Gas Dispersion Model

Benjamin & Rothkrantz (2007) developed an automated crisis simulator involving a toxic cloud incident. In particular, the project was aimed at exploring evacuation plans in the area where a chemical plant was located. The study took a chemical complex around the harbor of Rotterdam as the study case. At time there could be an explosion in one of the plants, the simulator showed that a toxic cloud could be spreading in the direction of the city and the nearby University of Rotterdam.

The simulator applied a Gaussian-Plume-model (GPM) for modeling light and neutral gases (see Figure 2(a)). Figure 2(b) shows the concentrations of a gas cloud are being spread across an area, on a height of 2 meters (the approximated height of people walking in the field).

The actual model comes down to a calculation of a concentration on a location (x, y, z) in which x, y and z are relative coordinates from the dispersion source in the sense that the x-axis is the wind direction with the origin in the dispersion source point and the cross wind (side wind) direction is the y-axis, also with the origin at the dispersion source. The concentration on a time t for (t > 0) is:

\[
\nu(x, y, z, t) = \frac{m}{(2\pi)^{3/2}\sigma_x \sigma_y \sigma_z} \cdot v_x \cdot v_y \cdot v_z \exp\left(-\frac{\left(x - u_w t\right)^2}{2\left(\sigma_x^2\right)}\right) \exp\left(-\frac{y^2}{2\left(\sigma_y^2\right)}\right)
\]

\[
v_x = \exp\left(-\frac{\left(x - u_w t\right)^2}{2\left(\sigma_x^2\right)}\right)
\]

\[
v_y = \exp\left(-\frac{y^2}{2\left(\sigma_y^2\right)}\right)
\]
\[ v_x = \exp \left( -\frac{(x-h)^2}{2\sigma_x^2} \right) + \exp \left( -\frac{(x+h)^2}{2\sigma_x^2} \right) \]

In this equation, \( m \) is the released mass, \( u_w \) is the wind speed and \( h \) is the source height. \( \sigma_x, \sigma_y, \) and \( \sigma_z \), the dispersion parameters, depend on the wind speed \( u_w \) and \( t \) the time of year.

The simulator used the gas dispersion model to predict the path of the gas dispersion plume. It included features for plotting the gas dispersion on top of a map of a certain area and for calculating a reasonable approximation of the actual gas concentration on a given point \((x, y, z)\) in a time extent of an hour. For the routing system, the calculated affected regions can be used as an input to compute alternative safe routes.

**DYNAMIC ROUTING ALGORITHM**

The algorithm uses an annotated graph to represent the street network of a city. The edges correspond with the streets and the nodes with the intersection of the streets. The edges are annotated with the traveling time between neighboring nodes. The Dijkstra algorithm, then, is used to compute the shortest traveling time between nodes. The original Dijkstra algorithm calculates the matrix of the traveling time between nodes and returns a shortest route from a start node to a destination node. The adapted version takes into account the dynamic situations where some nodes may not be accessible anymore and the traveling time between remaining nodes need to be recomputed by predicting the (dis-) availability of all nodes in the future.

Figure 3 shows the illustration of the algorithm. At each cycle, the algorithm calculates multiple versions of a particular annotated graph. Each graph represents the situation at a particular time \( t \). The dynamic changes at \( t \) onto the graph(\( t \)). i.e. graph(\( t \)), and changing the weight of the edges. By this way, we add the time parameter as a new dimension to the original Dijkstra algorithm. The example shows a space graph represented by nodes \{A, B, C, D, and E\} that is repeated for three time interval \( t=0..2 \) with interval 5 minutes. All edges that connect the nodes at initial time \( t=0 \) is colored black, which means that all routes are available. The illustration shows that at \( t=0 \) a toxic cloud is beginning to come from the left. By calculating the possible gas dispersion path, the system can predict that at \( t=0 \) the node B becomes and \( t=0.05 \) the node D becomes dangerous areas. Assuming that E is the destination, the routing algorithm calculates the shortest route at \( t=0 \) and results the path A-B-D-E. However, since the links between the nodes A and B and the nodes B and D are not available, the shortest route at \( t=0.05 \) is the nodes...
A-C-D-E. Then, the shortest route at $t=00:10$ is the nodes A-E, since the node D becomes dangerous and the links between the nodes C and D and the nodes D and E are not available. This example shows that at regular time steps, i.e., 5 minutes, the shortest routes are computed via the links and using the eventually updated links in the graph. The final output is the path A-E.

**Algorithm:**

```plaintext
GetTravelTime(i, j, t_i, av_{ij}(t))
// This procedure implements the travel time estimation
// methodology for two successive nodes i and j
{ i is current node, j is successor node of i }
{ t_i is current time at node i }
{ t is the last update time }
{ av_{ij}(t) is the availability of edge_{ij} at time t }
{ I_k is the interval }
L ← d(i, j) { d(i, j) is the travel time between i and j }
K ← 0 { start the iteration with the first time interval of that road }
WHILE t_i is one of I_k DO
    t_i ← t_i + L/av_{ij}(I_k)
    K ← K+1
END WHILE
Return t_i
```

**DynamicDijkstra(V, s):**

(V is the set of nodes in a graph)
{s is the source node}

Initialize (V, s) with the maximum value of the travel time between nodes
S ← ∅ { Set of all visited nodes }
Q ← CreateEmptyList() { Set of all unvisited nodes }
WHILE Q≠∅ DO
    i ← ExtractMin(Q) { i is a node from Q, not in S }
    AddToSet(S, i) { adding i to S }
    FOR all adjacent vertices of i DO
        { Determine if current route to j via i is faster }
        alt ← GetTravelTime(i, j, t_i, av_{ij}(t))
        IF alt < d(i, j) THEN
            d(i, j) ← alt { the route to j via i is faster }
            previous[j] ← i { the previous node on the shortest path to j is i }
        END IF
    END FOR
END WHILE
```

Based on this illustration, the function `GetTravelTime` algorithm considers that updates of the actual availability of an edge $av_{ij}$ are received at every time interval $\Delta_t$. The value of $av_{ij}$ becomes infinity if the node is not accessible anymore (i.e., become dangerous). The cost of every edge is the travel time, computed in the algorithm below, and not the road distance as in the original Dijkstra.
The function GetTravelTime is used in the DynamicDijkstra algorithm to calculate a new travel time between two nodes and to determine whether the current computed route is faster.

EXPERIMENT

We conducted two experiments in the laboratory setting to validate our algorithm. The first was using a Lego robot that was moving through a maze of nodes. The second was using a crisis simulator.

The following sections describe both experiments sequentially.

The Lego-Robot Experiment

We used a maze of black lines to represent the road network (Figure 4 (a)). The task of the robot was to walk from a source node to a destination node following the black line. We simulated dangerous areas by covering the black line with a piece of white paper during runtime at random time so that the robot was unable to track the line.

A routing system with the Dynamic Dijkstra algorithm guided the robot to the destination. The system received information about the situations from a webcam attached above the maze. It was able to survey the maze and detect the blocked lines. Figure 4 (b) shows the representation of the maze in the internal memory: the blue node shows the position of the robot, the green lines are safe routes, and the red lines are dangerous routes.

At every crossing, the robot had to take a decision: turning left, turning right or straightforward. Without the guidance of the routing system, the robot was able to reach the destination by trial and error. If a line were covered, the robot would return and follow previous route.

We found that with the guidance of the routing system, the traveling time was reduced significantly. The dynamic routing algorithm was able to compute the shortest path from the source node without entering the blocked line. We also found that the traveling time depended on how many times and how fast the maze had been changed.

Figure 4. (a) Maze of black lines and line tracking robot and (b) representation of the maze in the computer memory.
The Crisis Simulator Experiment

We developed a synthetic environment of the city of Rotterdam (Tatomir et al., 2009). On this environment, we simulated a disaster caused by a toxic cloud applying the gas dispersion model to create situations that made some road hazardous and became disabled. To represent the affected people, we populated the roads with a fixed amount of virtual agents that were created at random positions. They moved to a certain destination that represents a safe area. The lines represented the road network and the dots represented the virtual agents (Figure 5).

Figure 5. Simulated environment applying the gas dispersion model.

In the experiment, we created a scenario that there was an explosion in one of the chemical plants in the west side of the city. The explosion had created fire and toxic cloud. The toxic cloud was spread in the course of the time. Due to this, many roads were blocked successively. The task of the agents was to flee to one destination without entering dangerous roads.

Figure 6. Average traveling time with the dynamic Dijkstra algorithm to a safe area: (a) with disabling a link and (b) without disabling a link. (c) Average traveling time without any dynamic routing.

At the start of each session of the experiment, we observed that the traveling time was increasing (see Figure 6). When the first agents arrived at their safe area the traveling time converged to a maximal value. If a link were disabled, we observed...
a sharp increase in the total traveling time (see Figure 6(a)). At this point, the resulted route of the routing algorithm might cause some agents to return or start a new possible longer route. After some time, all agents were routed in an optimal way. Without a dynamic routing we observed a continuous increase in traveling time (see Figure 6(c)). This was caused by the fact that the agents did not have optimal routes to their safe area. If one route were blocked they would try to find a route around the blocked road.

CONCLUSION

Continuing previous work on developing disaster-response technology, we developed a routing algorithm based on the Dijkstra shortest path algorithm. The algorithm takes into account the change of available routes coping the non-deterministic environments due to the destruction of the disaster. We have tested the algorithm both in simulated environments. The algorithm is able to compute the safe and fastest route. Future work includes testing the algorithm in the real-life situation.

To be useful, the algorithm needs to be applied in a system that is able to gather up-to-date information about situations in the disaster areas. Therefore, we aim at developing a dynamic routing system on a smartphones. The input of the system is icon-based messages of observational reports. A collaborative communication network will share the reports among people in the affected area. The routing system will compute the shortest routes to safe areas as well as provide information for others. This yields further questions: (1) whether or not affected people in disaster situation are willing to report their observation using the icon-based messages and (2) how to validate the reliability of the observation reports.

REFERENCES
