Path Planning for First Responders in the Presence of Moving Obstacles With Uncertain Boundaries

Zhiyong Wang, Sisi Zlatanova, and Peter van Oosterom

***Abstract*— In this paper, we study path planning for first responders in the presence of uncertain moving obstacles. To support the path planning, in our research we use hazard simulation to provide the predicted information of moving obstacles. A major problem in using hazard simulation is that the simulation results may involve uncertainty due to model errors or noise in the real measurements. To address this problem, we provide an approach to handle the uncertainty in the information of moving obstacles, and apply it to the case of toxic plumes. Our contribution consists of two parts: 1) a spatial data model that supports the representation of uncertain obstacles from hazard simulations and their influence on the road network and 2) a modified A* algorithm that can deal with the uncertainty and generate fast and safe routes passing though the obstacles. The experimental results show the routing capability of our approach and its potential for the application to real disasters.**

***Index Terms*— Data model, route planning, algorithm, uncertain moving obstacles.**

I. Introduction

During disaster responses, there is a great need to support navigation for first responders in the presence of moving obstacles [17], [20]. When disasters happen, different kinds of moving obstacles (e.g., fires, plumes, floods) can be caused by the hazards in the infrastructure, and block parts of the road network. To help responders fast and safely reach their points of interest, hazard model and simulations can be employed to provide the information of the obstacles (e.g., location, shape, speed), offering a promising way to support navigation during disasters [1], [9], [22].

One of the challenging issues of using the hazard simulation is the uncertainty in the simulation results. A number of factors or conditions, such as model errors, uncertainties in real measurements, can influence hazard simulation process, causing generation of uncertain results from the hazard models. For example, in prediction of forest fires, many factors can be involved in the fire simulation and make it difficult to predict fire-fronts. They could either be randomness in weather conditions, such as winds, precipitation, and humidity, or errors in the models, like terrain model and land use model.

Manuscript received April 24, 2016; revised September 9, 2016; accepted November 16, 2016. Date of publication January 2, 2017; date of current version July 31, 2017. The Associate Editor for this paper was L. M. Bergasa. *(Corresponding author: Zhiyong Wang.)*

The authors are with the Faculty of Architecture and the Built Environment, Delft University of Technology, 2628 BL Delft, The Netherlands (e-mail: wzy19840102@163.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2016.2634079

Another challenging issue is to find fast and safe routes during disasters. Although conservatively avoiding the obstacles can guarantee that responders can reach destinations safely, it would slow down the response process by wasting much time on traveling. Because the responders usually have protective equipment which allows them to pass through certain levels of hazards, it is also very important to investigate how to find routes that allow responders to pass through the obstacles but have limited risk.

To our knowledge, there are few works that investigate the routing for first responders in the presence of uncertain moving obstacles. In the GIS field, some research efforts have been directed to the path planning among moving obstacles. Visser [19] develops an obstacle avoiding routing algorithm that can incorporate the dynamic blocks caused by plumes. Using a flood model, Liu *et al.* [8] investigate the routing under flood disasters, and propose an adaptive routing algorithm which can take into account the effect of water depth on walking speed. Chitumalla *et al.* [1] present an application that uses the forecast information of plumes in the near future in the routing and provides navigation services taking blocked areas or streets into account. Wang *et al.* [22] propose a data model and an extended A* algorithm which can support path planning in the presence of forest fires. However, the above path planners assume that the shape and the size of the obstacles are accurately defined and completely certain, which limits their applications for real disasters. In the past years, there have also been some research works that use agent technology to assist path planning during disasters. Rahman and Mahmood [15] present an ant-based multi-agent system to provide feasible routes for building evacuation, considering the physical constrains of obstacles. Similarly, Forcael *et al.* [2] apply the ant colony optimization model to find safe evacuation routes in the case of Tsunamis. Due to lack of consideration of the uncertainty and complexity of environments affected by hazards, these agent-based route planning systems have serious limitations in dealing with the uncertain information of the obstacles during disasters.

In robotics some researchers have studied the path planning among uncertain obstacles, addressing issues caused by the uncertainty in real time observation of obstacles from sensors. Ok *et al.* [13] develop a path planner called Voronoi Uncertainty Fields, which uses Voronoi diagrams and potential fields to deal with map uncertainties. Neumany and Likhachevy [12] design a generalization to the PPCP (Probabilistic Planning with Clear Preferences) algorithm,

which allows a robot to reason about uncertainty in the trajectories of dynamic obstacles. Sonti *et al.* [16] present a grid-based path planning algorithm using probabilistic finite state automata (PFSA), and address the routing problems in the presence of dynamic obstacles with stochastic motion models. For navigating robots through large environments with thousands of uncertain dynamic obstacles, Neuman and Stentz [11] develop an anytime receding horizon technique, which is built on a dynamic programming approach with a heuristic search method. The path planners mentioned above allow robots generate routes based on the imperfect information of the obstacles, and even take into account the uncertainty in the future possible locations of obstacles. However, while focusing on the free space environment, their approaches handle the obstacles (e.g., humans, robots) that can be simply represented by points or cells which have uncertain locations, and are not suited to deal with the obstacles like fires, plumes, which can affect large areas with road networks and also have uncertainty in both geometries and properties (e.g., the concentration of plumes). Moreover, because of the physical constraints of robots, these research studies mainly focus on avoiding obstacles. Nevertheless, during the disasters, the responders can use equipment (e.g., masks) that can protect them from being affected by hazards. This protective equipment enables the responders to pass through certain obstacles to deliver their emergency services instead of conservatively avoiding the obstacles. Hence, there is also a necessity for taking into account the profile of the responders to generate routes that are safe enough for them to pass though the obstacles. However, as far as we know, not many attempts have been made to address these path planning problems.

In this paper, we study the path planning in the presence of moving obstacles with uncertain boundaries. We use hazard simulations to predict the movement of hazards, and provide an approach to handle the uncertainty involved in the results generated from hazard models. Our approach is composed of the following two parts: 1) a data model, which is built on the earlier developments proposed in [22], and introduces a set of new constructs (class, attribute, and association) to support representation of the uncertain obstacles generated from hazard simulations; 2) a routing algorithm, which uses the uncertainty information of obstacles to generate the safe routes through the moving obstacles. The proposed approach is generic and can also be applied to different types of hazards (e.g., floods, plumes). In this paper, we mainly consider its application to the case of plumes. The remaining part of the paper will be as follows: in Section II, we describe a system architecture which can provide routing service among uncertain moving obstacles. Section III presents our designed data model. In Section IV, we illustrate the routing algorithm which is developed based on the A* routing algorithm. Section V gives some results of the application of the data model and algorithm. This paper is ended with some conclusions and future works in Section VI.

II. The System Architecture

To support routing among uncertain moving obstacles, we propose a system architecture which can deal with the uncertainties in the hazard simulations. In this paper, we mainly consider the moving obstacles that have uncertain boundaries (as illustrated in Figure 1). As shown in Figure 2, the proposed system architecture consists of three main components: hazard models, a geo-database management system (geo-DBMS), and a multi-agent based navigation system. When disasters happen, the sensor data is collected from the field, and used to drive the hazard models to predict the movement of the obstacles. Because the uncertainties in hazard simulations are commonly related to random variables or stochastic parameters, hazard model and simulations that employ Monte Carlo (MC) method [3], [4], [23] are selected and used to quantify the uncertainties. The hazard model is executed for a predetermined number of times, generating the forecasted information of hazards during a specific time period. We collect the data generated from each hazard simulation for each timestamp, and transform it into a set of obstacle polygons. On the basis of these polygons, we compute the risk probability of being affected by the uncertain obstacles. The more polygons come over certain areas, the higher probability that the obstacles will affect the areas. In this study, we use the geo-DBMS to store the results from hazard simulations, and apply the agent technology to support the routing. Different types of agents are developed to automate the spatial data processing and analysis of the simulation results, according to the framework designed in [21]. The agent system makes a direct connection to the geo-DBMS, and fetches the data of hazards. Based on the risk probability, it determines the states of each road, considering the profile of first responders. Using the state information of road networks, the routing algorithm is performed and generates safe and fast routes for the responders.

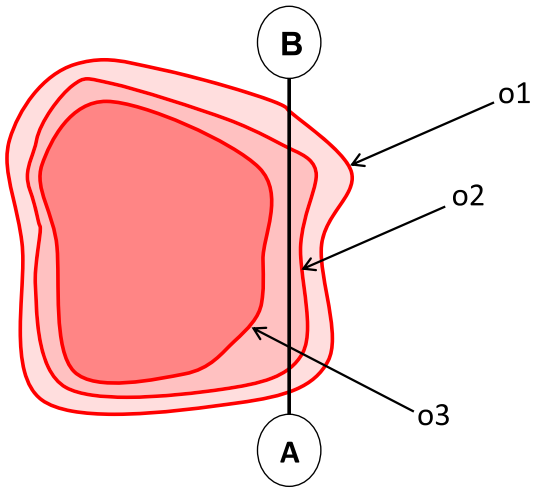


Fig. 1. A set of obstacle polygons (i.e., $o1$, $o2$ and $o3$) representing smoke plumes. We assume that the obstacles have the same property (i.e., the same concentration of plumes). The obstacles occur at the same simulation time, but have different locations and boundaries. Each obstacle corresponds to a specific execution of the plume model. The road segment A_B is affected by obstacles $o1$ and $o2$, and not affected by obstacle $o3$.

III. Data Model Representing Uncertain Moving Obstacles

In this section, we present a data model, which is used to help structure the routing-related data in the geo-DBMS.

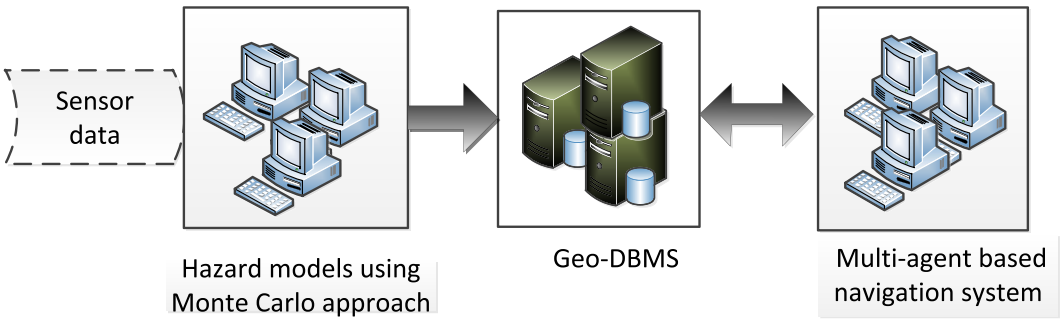


Fig. 2. The proposed system architecture.

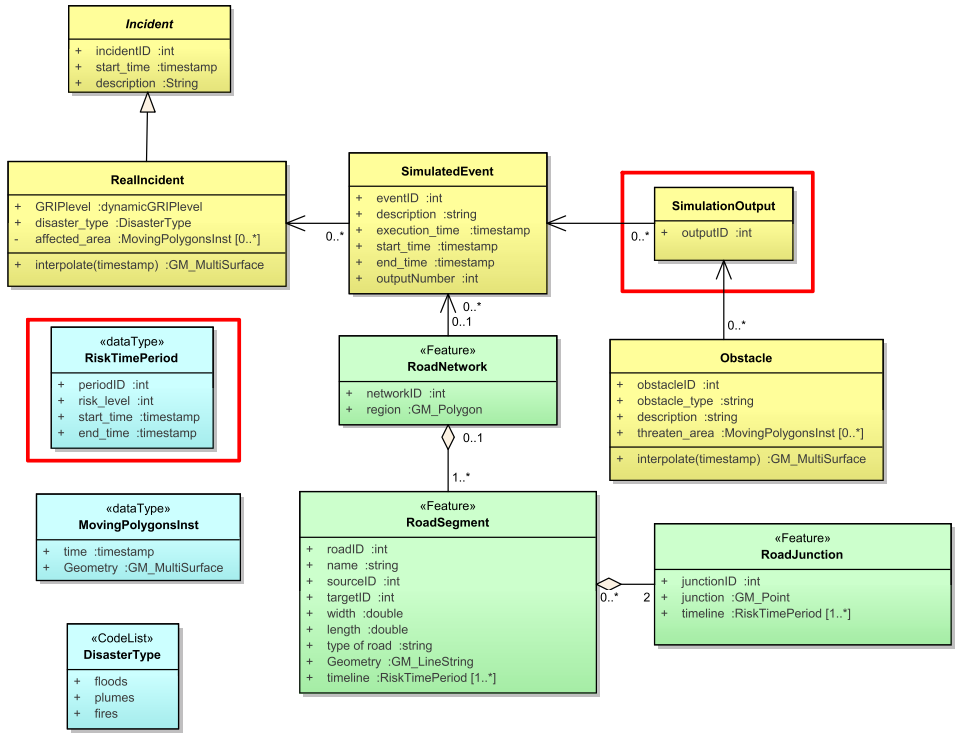


Fig. 3. The core of the data model for representing the uncertain moving obstacles.

We build our model based on the earlier works in [22], and extend it with a set of new constructs (class, attribute, and association) to describe and quantify these uncertainties from the hazard simulation models. Figure 3 shows the core of the designed model. In this paper, we introduce the concept of levels to represent the state of the environment affected by the uncertain moving obstacles, instead of using binary values (i.e., closed and open).

In our data model, the class SimulatedEvent is linked with RealIncident and describes the simulations that predict the movement of hazards caused by incidents. Another new class SimulationOutput (highlighted in red in Figure 3) is introduced to differentiate and store multiple simulation results. It represents a specific execution of the simulation model, and links with a set of output data of Obstacles. A SimulationEvent can be associated with a number of SimulationOutput. Note that the input information for the simulation model is not included in the model, because the input variables vary in different simulation models and are out of the scope of this research. But for a specific simulation model, this information can be easily organized in a separate class which can be linked to the SimulationOutput class.

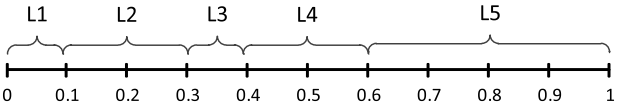
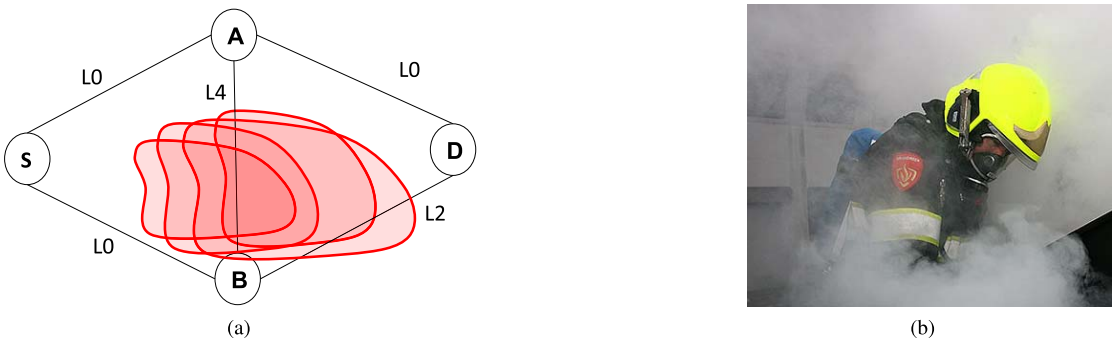


Fig. 4. Defining of risk levels based on the risk probability.

The *risk level* is the main concept introduced to manage data about the uncertainty of components of the road network. In this paper, the status of road segments and junctions affected by the moving obstacles with uncertain boundaries is not represented by their availabilities (i.e., closed and open), but is described by a set of levels of risk. In this study we use the probability approach to quantify the uncertainty, and define the *risk level* according to the risk probability of intersecting with obstacle polygons. In our model, we do not explicitly store this risk probability, but discretize it by breaking the interval $[0, 1]$ into a finite number of smaller intervals. Each *risk level* corresponds to a specific risk probability interval. Take Figure 4 as an example, *risk level* $= L1$ corresponds to a range



(a) (b)

Fig. 5. Navigation for a first responders with a gas mask. (a) A graph representing a road network affected by smoke plumes (in red polygons). Each polygon corresponds to a specific execution of the plume simulation. The risk level of each road segment at a certain timestamp is shown adjacent to the associated edge. (b) A responder wearing a gas mask. We assume that the responder has to go from S to D, and He is allowed to pass through the roads with the risk levels L0 and L2, but should avoid the roads with the risk levels L4.

of risk probability $[0, 0.1)$, *risk level* $= L2$ corresponds to a range of risk probability $[0.1, 0.3)$, and so on. The larger the number in the *risk level* is, the more chances of exposure to the obstacles would be encountered. Using the concept *risk level*, the proposed model allows for a multi-state representation of the affected environment.

To determine the *risk level*, we adopt the following procedure, which couples the Monte Carlo method to obtain the risk probability of being affected by the uncertain obstacles:

Step 1 Generate samples from N_s realizations of the given random variables.

Step 2 Input these samples to run the simulation model, and generate N_s simulation results.

Step 3 For each simulation result, determine whether the road segments and junctions are affected by checking if they intersect with the obstacle polygons.

Step 4 Count the number of times of being affected, i.e., M_a.

Step 5 Calculate the fraction of being affected by moving obstacles at a given time point, using the following formula:

$$P_b = M_a / N_s \tag{1}$$

This fraction P_b is interpreted as an indicator of the likelihood of being affected by the obstacles. In our research, we use it as the risk probability to identify the *risk level* we introduced earlier.

A new data type RiskTimePeriod is created in the model, as highlighted in red in Figure 3. It stores the information of *risk level* and describes the time period over which the road is at the predicted risk level. Attributes of RiskTimePeriod are: periodID, which is the identifier of this time period; risk_level, which indicate the *risk level* during this time period; start_time, which is the starting time of the time period; end_time, which is the ending time of this time period. Using this data type, a new attribute timeline is added in the classes RoadSegment and RoadJunction to maintain information of the status of the road network at different time periods. In the following section, we will illustrate how to use the *risk levels* to calculate the safe routes in the presence of uncertain moving obstacles.

IV. Routing Algorithm

Given the predicted information of a road network affected by uncertain moving obstacles, we formulate the path planning problem as follows. Here we assume that a response unit has to move from a source to a destination in the road network, and departs from a given start time. The responders have their preferences, i.e., they can pass through some moving obstacles if the risk of the moving obstacles they encounter is below a certain level, depending on the available protective equipment. Besides, they also have time constrains, i.e., they have a limited amount of time for moving through the obstacles. Figure 5 shows an example of guiding a responder who wears a gas mask. The responder has to pass through a road network affected by smoke plumes (as shown in Figure 5a). He can move through some roads with certain risk levels, but his movement is limited by the amount of oxygen that can only be used for a certain period of time (see Figure 5b). Under the above assumptions, the problem is to compute a feasible route to the destination, taking into account the *risk level* of road segments and junctions as well as the time of passing through the obstacles.

To address the problem mentioned above, in this section we present a Moving Obstacle Avoiding A* algorithm, following the basic principles of A* algorithm proposed by Hart *et al.* [7]. Our algorithm can deal with the uncertainty of the moving obstacles, and is named MOAAstar/Uncertainty. Extending the concept of safe intervals as used by [10] and [14], in MOAAstar/Uncertainty we distinguish three types of intervals: *full-safe*, *partial-safe* and *non-safe intervals*. These intervals are defined by the users based on the *risk level* that we introduced in the data model (Section III) to describe the status of road segments and junctions. In the algorithm we use a continuous risk function to accumulate the time of moving along the *partial-safe* roads, and add it into the state as a new state variable (represented by r), modeling the risk value as an additional dimension. Moreover we introduce a new parameter, r_{max}, which indicates the maximal amount time that the responders are allowed to pass through *partial-safe* roads. The objective of the algorithm is to try to minimize the arrival time to the destination, constraining the time of passing through *partial-safe* roads (i.e., $r \leq r_{max}$)

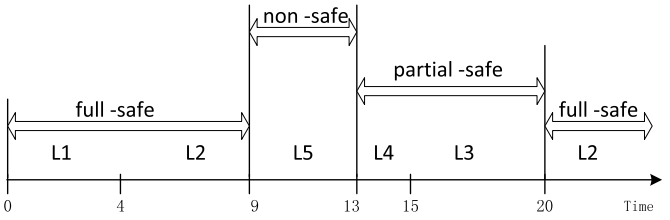


Fig. 6. A timeline consisting of three types of intervals defined based on the risk levels.

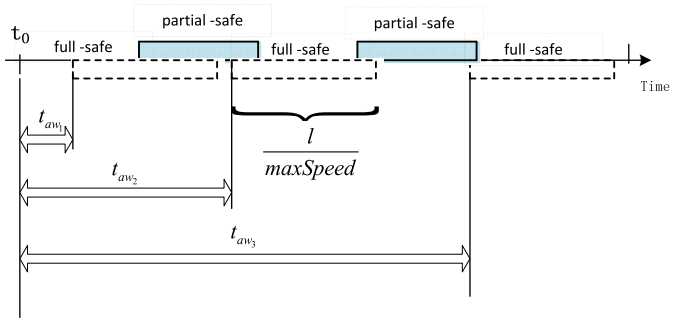


Fig. 7. An example of different waiting times in passing through an edge.

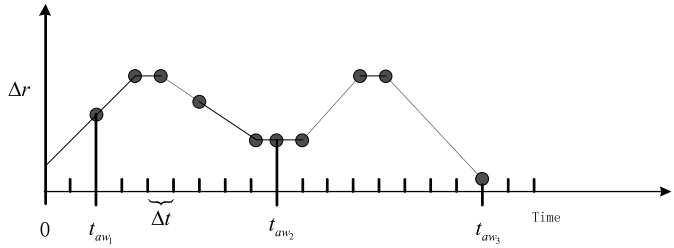


Fig. 8. The addition of the risk with different waiting times.

and avoiding the *non-safe* roads. Because of the addition of the risk function, the dimensionality of the search space is also increased, which incurs more computational cost. For addressing this problem, a special technique is required in the algorithm to limit the search area.

A. Defining Safe Intervals

Based on the risk_level in the proposed data model (Section III), we divide the time dimension of the hazard simulation into different groups of intervals. Specifically, the RiskTimePeriods are categorized into the three types of intervals that are mentioned earlier: *full-safe intervals* which are free to pass through, and *partial-safe intervals* that can still be passed through, but have some risks and *non-safe* intervals which are not allowed to pass through. Accordingly, the corresponding *risk levels* are called *full-safe levels*, *partial-safe levels*, and *non-safe levels* respectively. In our approach, the responders are involved in defining the safe intervals for the search space. They classify RiskTimePeriods into the three defined types of intervals, based on their preferences. According to different profiles of the responders, each RiskTimePeriod would fall into a different type of intervals.

More formally, the list of RiskTimePeriods, which is stored in the attribute timeline for each road segment and junction, is divided into the following three temporal sets

MOAAstar/Uncertainty

```
Initialize source S, destination D
initialize r_max, maxSpeed, departureTime
openSet ⟵ ∅, closedSet ⟵ ∅
generate source state s_source
insert s_source in openSet
while openSet is not empty do
    s ⟵ the state in openSet having the lowest f value
    if node(s) = D then
        return the path from source S to destination D
    end if
    remove s from openSet
    insert s to closedSet
    successors ⟵ generateSuccessors(s)
    for each successor s' in successors do
        if ¬ checkDominated(s') then
            insert s' into openSet
        end if
    end for
end while
return no-path
```

Fig. 9. The main structure of the algorithm MOAAstar/Uncertainty.

according to the user profile: $(p_1^\alpha, \ldots, p_q^\alpha, \ldots, p_Q^\alpha)$, $(p_1^\beta, \ldots, p_m^\beta, \ldots, p_M^\beta)$, $(p_1^\gamma, \ldots, p_u^\gamma, \ldots, p_U^\gamma)$, where p_q^α represents the interval that is fully safe; p_m^β is the interval that is partially safe; p_u^γ indicates the interval that is not safe. $p_q^\alpha = (t_{oq}^\alpha, t_{cq}^\alpha)$, $t_{oq}^\alpha < t_{cq}^\alpha$, t_{oq}^α is the start time of the full-safe interval p_q^α, t_{cq}^α denotes the end time of p_q^α, and Q is the total number of the *full-safe intervals*. Similar definitions hold for the partial-safe interval p_m^β and the non-safe interval p_u^γ. M is the total number of the *partial-safe intervals*, and U is the total number of the *non-safe intervals*. By grouping together the *partial-safe intervals* and *full-safe intervals*, we can obtain the open intervals, $(p_1, \ldots, p_k, \ldots, p_K)$, which are the intervals that allow responders to pass through the roads. For example, for a user who defines *risk level* as follows: *full-safe levels*={$L1, L2$}, *partial-safe levels*={$L3, L4$}, and *non-safe levels* = {$L5$}, a timeline, as shown in Figure 6, can be converted into a list of *full-safe intervals* $([0, 9], [20, +\infty])$, a list of *partial-safe intervals* $([13, 20])$, and a list of open intervals $([0, 9], [13, +\infty])$. It should be noted that we do not consider the *non-safe* intervals in the routing as they are excluded from the search space.

B. Defining State and State Dominance

In the algorithm, we use the open intervals to define the state for building the search space. However, to support path planning with risk function that accumulates the time of moving in the *partial-safe intervals*, we explicitly include the risk function as part of the state space. Formally, a state is defined as follows: $s = (x, p', p'', r)$, where x is the node associated with the state; p' corresponds to one of the open intervals of node x, i.e., p_k; p'' is either a *full-safe interval* p_q^α or a *partial-safe interval* p_m^β; r is the risk variable representing the

generateSuccessors(s)

```
x ⟵ node(s), [t^x_ok, t^x_ck] ⟵ the open interval associated with s
new_states ⟵ ∅
for each neighbor y of x do
  for each open interval i of edge xy do
    if [g(s), t^x_ck] ∩ [t^xy_oi, t^xy_ci] = ∅ then
      continue
    end if
    earliest_start_time ⟵ max(g(s), t^xy_oi)
    derive the maximum waiting time t^max_aw
    t_aw ⟵ 0
    while t_aw <= t^max_aw do
      start_time ⟵ earliest_start_time + t_aw
      if start_time > t^xy_ci then
        break;
      end if
      arrival_time ⟵ start_time + l_xy/maxSpeed
      calculate the incremental risk Δr
      if arrival_time < t^xy_ci and r(s) + Δr <= r_max then
        generate state s' of node y
        t_w(s, s') ⟵ start_time - g(s)
        if s' is in closedSet then
          continue
        end if
        if s' is in openSet then
          if arrival_time < g(s') then
            g(s') ⟵ arrival_time
            f(s') ⟵ g(s') + h(s')
          end if
        else
          g(s') ⟵ arrival_time
          f(s') ⟵ g(s') + h(s')
          insert s' into new_states
        end if
      end if
      if ¬ checkIncrement(start_time, xy) then
        break;
      end if
      increase t_aw with the time step Δt
    end while
  end for
end for
return new_states
```

Fig. 10. The function for generating successors, generateSuccessors(s).

checkDominated(s)

```
for each ŝ in openSet ∪ closedSet do
  if node(ŝ) = node(s) and p'(ŝ) = p'(s) and p''(ŝ) = p''(s) then
    if s is dominated by ŝ then
      return true
    end if
  end if
  return false
end for
```

Fig. 11. The function for checking if a state s is dominated, checkDominated(s).

accumulated time of passing through the *partial-safe interval* of roads. In this way, we can easily see how much time the responder has been going through the *partial-safe* obstacles, given a state s. As risk value r is obtained from a continuous

checkIncrement(start_time, xy)

```
p^{xy,α}_Q ⟵ the last full-safe interval of edge xy
p^{xy,β}_M ⟵ the last partial-safe interval of edge xy
if start_time is within p^{xy,α}_Q and the end time of p^{xy,α}_Q = inf then
  return false
end if
if start_time is within p^{xy,β}_M and the end time of p^{xy,β}_M = inf then
  return false
end if
return true
```

Fig. 12. The function for checking if the waiting time increment should be stopped, given the start_time and edge xy, checkIncrement(start_time, xy).

function of time, for a given node x within certain intervals p' and p'', there can be multiples states, all corresponding to different risk values.

Because the dimensionality of our search problem is increased by addition of the continuous risk function into the state, the search space is also enlarged, which results in more computing time and space needed for the algorithm to search for feasible paths. In this study, a state dominance relationship is defined and applied to reduce the search space. State dominance has been used in many planning problems to limit the search space, and thus to improve the computation speed [5], [6]. The idea of state dominance is that if a state s is dominated by another state s', the solution obtained from s can not be better than the solution found from s', which means that state s would unlikely contribute to the optimal solution. Using the state dominance, we can facilitate the search process by identifying the dominated states and pruning them from further expansions in the search, without compromising the optimality of the solution.

For our path planning with the risk function, we derive a state relationship based on the state defined above. In the algorithm, we aim to minimize the travel cost for the responders, while limiting the total risk that is accumulated in passing through *partial-safe* roads. In this sense, a state with less travel cost and lower risk would dominate the one with more travel cost and higher risk. More specifically, given two states s and s', which refer to the same node within the same intervals, but differ on the risk values: $s_u = \{x, p', p'', r_u\}$, $s_v = \{x, p', p'', r_v\}$, if ($g(s_u) <= g(s_v)$ and $r_u < r_v$) or ($g(s_u) < g(s_v)$ and $r_u <= r_v$), then we say that state s_u dominates state s_v. For state s, $g(s)$ represents the least cost found so far from the source to s; r denotes the amount of risk that has been accumulated from the source to s.

C. Discretizing the Search Space

When the vehicle moves along an edge that has multiple *full and partial safe intervals*, the accumulated risk varies with the starting time from one node of the edge to the other. For example, as shown in Figure 7, the vehicle can depart from the earliest start time t_0, and moves at the maximum speed. The dash lines have the equal length of time, and represent the time periods that the vehicle needs to travel through the edge of a given length. Different waiting times, t_{aw_1}, t_{aw_2}, t_{aw_3} , can be introduced and allow the vehicle to start its movement at

(a) (b) (c) (d)

Fig. 13. Snapshots of the movement of the plumes (in red polygons). (a) t=0 min. (b) t=3 min. (c) t=9 min. (d) t=18 min.

different times. The addition of the risk Δr is calculated as the sum of the time periods when the *partial-safe intervals* overlap the traveling period. It is easy to see that Δr is a value between $\Delta r^{min} = 0$ and $\Delta r^{max} = l/maxSpeed$. Figure 8 shows the accumulated risk values that correspond to the waiting times. As we can see, given different waiting times, the accumulated risk values are also different. Although waiting would make the responders take more time to arrive the destination, it may have some advantages in limiting the risks that the responders would encounter, and should be considered in the routing.

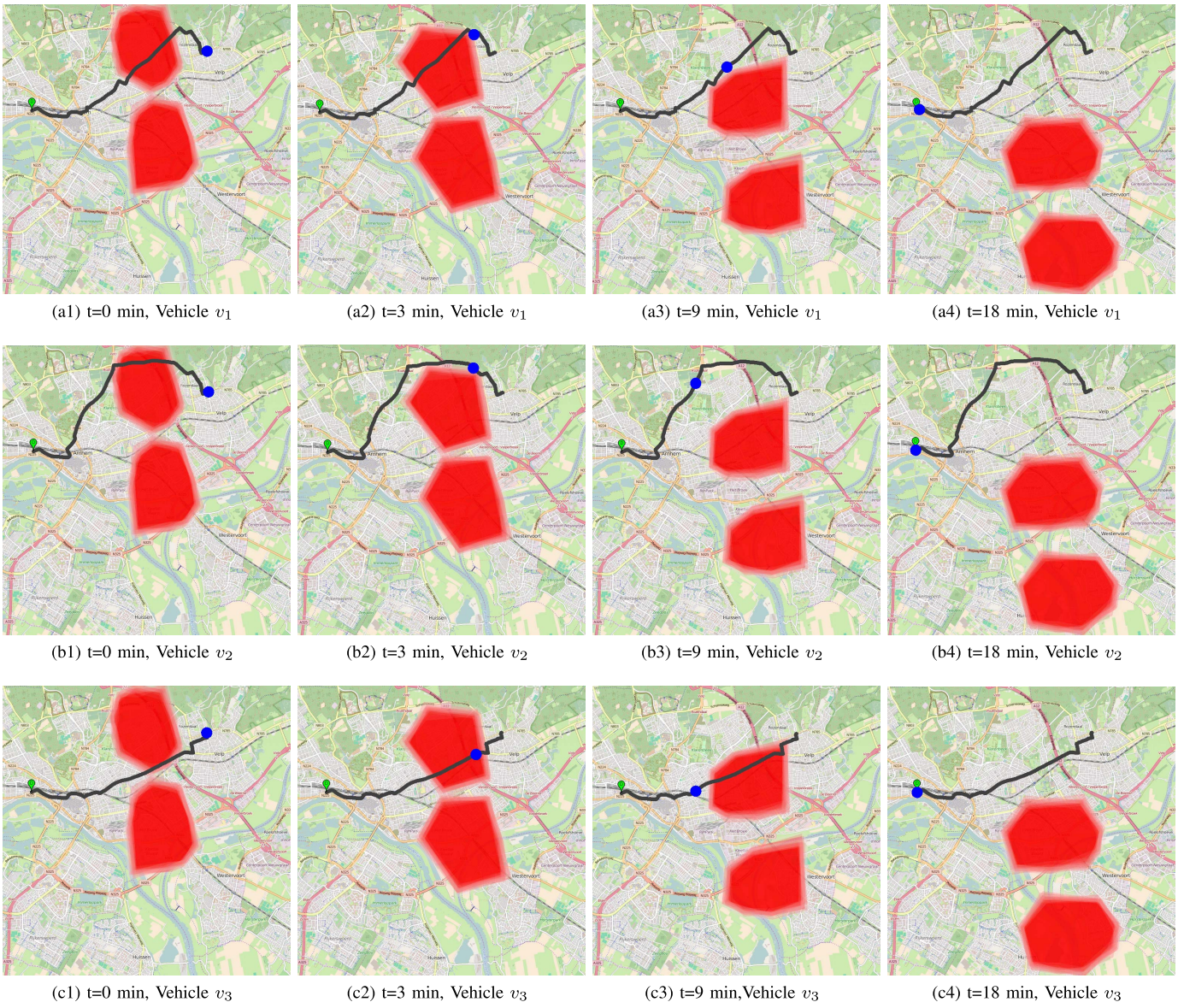
Following the above discussions, we introduce an additional waiting option that would lead to less risk, plus the waiting for the opening of edges. Because using contiguous timeseries to generate states would cause the size of the search space significantly enlarged during the search, we use a time discretization approach in the paper, inspired by the work of Van Den Berg and Overmars [18]. In the algorithm the waiting time series are discretized into small timesteps, Δt, to generate succeeding states for further expansions, as shown in Figure 8. We assume that the chosen time step Δt is small enough to be able to generate all possible states. Special constraints will be imposed on the waiting time, limiting the generation of successors.

D. Planning in the Defined Search Space

Figure 9 shows the main structure of our algorithm MOAAstar/Uncertainty. The algorithm starts with a given source and destination, the maximum speed and the departure time. Each state s in the algorithm is associated with a cost $f(s)$, which is computed as the sum of the actual cost $g(s)$ and the heuristic cost $h(s)$. When the state s with the lowest cost value is selected for further expansion, On line 13 (Figure 9) we use function $GenerateSuccessors(s)$ (Figure 10) to find successors of state s, considering all possible *full and partial-safe intervals* in transition from state s. On line 15 (see Figure 9), using the state dominance function defined in Figure 11, we iterate over all states in the successors to check if the successor is dominated by the states that have been found. We only insert the non-dominated successors into the open set for future expansion, which would reduce the search space for the path planner.

In generation of successors (Figure 10), the algorithm first examines each open interval $[t_{ci}^{xy}, t_{ci}^{xy}]$ of edge xy to see if it overlaps with $[g(s), t_{ck}^{x}]$ (line 5-7, Figure 10). Only the overlapped intervals are safe for the vehicle to pass through the edge, and will be considered. Then we use an incremental approach to generate possible successors, increasing the waiting time with the fixed time step. To derive the maximum waiting time, t_{aw}^{max}, that the vehicle can wait after the earliest start time (line 9, Figure 10), we distinguish *partial-safe interval* and *full-safe interval* for state s: 1). If s is within a *full-safe interval*, t_{aw}^{max} is set to the difference between the closed time of *full-safe interval* $t_{cq}^{x,\alpha}$ and the earliest start time; 2). If s is within a *partial-safe interval*, t_{aw}^{max} is set 0. This is because that waiting within the *partial-safe interval* does not provide the possibility of reducing the risk value of the successors, and would not be advantageous. The obtained t_{aw}^{max} is used as a limit on the increment of waiting time t_{aw} on line 11 (Figure 10).

In every loop, on line 13 (Figure 10) we first check the given waiting time to guarantee that the vehicle starts before the edge is closed. After that, we calculate the possible arrival time from node x to y (line 16, Figure 10), and estimate the risk increment, Δr, that would be accumulated during the travel through edge xy (line 17, Figure 10). If the vehicle can safely pass through the edge and arrives with a new risk lower than r_{max} (line 18, Figure 10), we generate a successor s', and compute the waiting time needed for transition from s to the state s' (line 19-20, Figure 10). As in the regular A* search, we ignore the state s' that is in the $closedset$, because it has been expanded in previous search (line 21-23, Figure 10). If the state is in the $openSet$ and the newly found path has a shorter time, we update the existing state with the new arrival time and the estimated traveling time $h(s')$ (line 24-28, Figure 10). The heuristic $h(s')$ is calculated based on Euclidean distance between node y and the destination D and the $maxSpeed$. The newly created state is inserted into the $openSet$ for further expansion (line 30-32, Figure 10). It is important to note that the closed time of the *full-safe interval* of state s (i.e., $t_{cq}^{x,\alpha}$) and the closed time of edge xy (i.e., t_{ci}^{xy}) could be infinity, which would cause an infinite loop in the algorithm. To prevent this, we adopt an additional function (Figure 12) to check if the increment of waiting time should be stopped, making the algorithm generate a finite set of successors. In Figure 7 and Figure 8, we have shown that given an earliest start time and a traveling period, the accumulated risk changes with the waiting time. In the following, we will prove that during the increment



(a1) t=0 min, Vehicle v_1 (a2) t=3 min, Vehicle v_1 (a3) t=9 min, Vehicle v_1 (a4) t=18 min, Vehicle v_1

(b1) t=0 min, Vehicle v_2 (b2) t=3 min, Vehicle v_2 (b3) t=9 min, Vehicle v_2 (b4) t=18 min, Vehicle v_2

(c1) t=0 min, Vehicle v_3 (c2) t=3 min, Vehicle v_3 (c3) t=9 min,Vehicle v_3 (c4) t=18 min, Vehicle v_3

Fig. 14. Snapshots of routes calculated for 3 vehicles (in blue circle) with different r_{max}. The vehicles have the same source and destination points. The shapes and positions of the obstacles (in polygons) are the same in the three simulations, and change as the vehicles are moving towards the destinations. (a) Vehicle v_1, $r_{max} = 0$ min. (b) Vehicle v_2, $r_{max} = 2$ min. (c) Vehicle v_3, $r_{max} = 7$ min.

of the waiting time t_{aw}, the generated states that do not meet the conditions on line 35 (Figure 10) will be dominated by a state generated earlier.

Theorem 1: If state s is within a full-safe interval that has an infinite end time, $t_{cq}^{x,\alpha} = inf$, and the considered open interval on edge xy also has an infinite end time, $t_{ci}^{xy} = inf$, there exists a waiting time t_{aw}^0, such that the states that are generated after t_{aw}^0 will be dominated by the state that corresponds to t_{aw}^0.

Proof: $t_{ci}^{xy} = inf$ implies that either the end time of the last full-safe interval, $t_{cQ}^{xy,\alpha}$, is infinity or the end time of the last partial-safe interval, $t_{cM}^{xy,\beta}$, is infinity. In the case that $t_{cQ}^{xy,\alpha} = inf$, we select the start time of the last full-safe interval as the start time of the vehicle, then get $\Delta r = 0$. We denote the corresponding waiting time by t_{aw}^0. For any $t_{aw} > t_{aw}^0$, the generated state s from t_{aw} would have a longer path, thus $g(s) > g(s_0)$ and $r(s) = r(s_0) = 0$. Based on our defined state dominance relationship, state s is dominated by s_0. Similarly, in the case that $t_{cM}^{xy,\beta} = inf$, we select the start time of the last partial-safe interval as the start time of the vehicle, and also represent the waiting time by t_{aw}^0. The obtained Δr is Δr^{max}. For any $t_{aw} > t_{aw}^0$, the generated state s from t_{aw} would have a larger path cost, thus $g(s) > g(s_0)$ and $r(s) = r(s_0) = \Delta r^{max}$. s is also dominated by s_0. ■

V. Case Study

In this study, the proposed data model (presented in Section III) has been implemented in the spatial Database Management System (DBMS) PostgreSQL with PostGIS extension (www.postgresql.org). In connection with

(a1) t=0 min, Vehicle v_1 (a2) t=3 min, Vehicle v_1 (a3) t=9 min, Vehicle v_1 (a4) t=18 min, Vehicle v_1

(b1) t=0 min, Vehicle v_2 (b2) t=3 min, Vehicle v_2 (b3) t=9 min, Vehicle v_2 (b4) t=18 min, Vehicle v_2

(c1) t=0 min, Vehicle v_3 (c2) t=3 min, Vehicle v_3 (c3) t=9 min, Vehicle v_3 (c4) t=18 min, Vehicle v_3

Fig. 15. Snapshots of routes calculated for 3 vehicles (in blue circle) with different categorization of risk levels. The vehicles have the same source and destination points. The shapes and positions of the obstacles (in polygons) are the same in the three simulations, and change as the vehicles are moving towards the destinations. (a) Vehicle v_1, *full-safe levels*={L0, L1, L2}, *partial-safe levels*={L3, L4, L5}, *non-safe levels* = $\emptyset$. (b) Vehicle v_2, *full-safe levels*={L0, L1}, *partial-safe levels*={L2, L3}, *non-safe levels* ={L4, L5}. (c) Vehicle v_3, *full-safe levels*={L0}, *partial-safe levels*={L1}, *non-safe levels* ={L2, L3, L4, L5}.

the database, a multi-agent based navigation system has been developed to support the path planning [21]. It uses the extended A* routing algorithm MOAAstar/Uncertainty (presented in Section IV) to generate paths in the environment affected by the uncertain moving obstacles. In this section, we test the system, using the road network dataset in Arnhem, The Netherlands. The road network is comprised of 13336 road segments and 11712 road junctions. We suppose that two moving toxic plumes are moving across the city. A group of datasets of polygons with timestamps have been created to simulate the movement of plumes (see Figure 13). At each timestamp, there are 50 polygons which are generated following a given normal distribution of positions. Our navigation system fetches the data of obstacles, and performs spatial analysis. By counting the polygons that intersect the road segments and junctions for each timestamp, the system computes the likelihood of being affected by the plumes. Here we define 6 levels of risk as follows: $L0$= $[0, 0.02)$, $L1$ = $[0.02, 0.05)$, $L2$= $[0.05, 0.1)$, $L3$= $[0.1, 0.3)$, $L4$ = $[0.3, 0.6)$, $L5$= $[0.6, 1]$. The information of risk levels of road segments and junctions is stored in the database according to the designed data model. We consider the following two scenarios: 1) The responders have the same preference but different time constraints (see Section V-A); 2) The responders have the same time constraint but different preferences (see Section V-B). The system generates the customized routes based on the profile of responders, and displays the calculated results on 2D maps.

A. Scenario 1: Navigation for Responders With Different Time Constraints

In this scenario, the system calculates the safe routes for 3 vehicles that have to go from the same source and destination points, given the same maximum speed 30 km/h

TABLE I

CALCULATED RESULTS CONSIDERING DIFFERENT TIME CONSTRAINTS r_{max}

Vehicle ID	r_{max} (mins)	Partial-safe range	Total Risk (mins)	Distance (km)	Total waiting time (mins)	Arrival time (min)
v_1	0	$[0.1, 1]$	0.0	7.5	2.6	18.2
v_2	2	$[0.1, 1]$	0.5	8.2	0.0	16.7
v_3	7	$[0.1, 1]$	6.0	6.6	0.0	14.8

Notes:
[1] Vehicles v_1, v_2 and v_3 have the same source and destination
[2] The routes are calculated by the algorithm, MOAAstar/Uncertainty, given the maximum speed of 30 km/h and a departure time t = 0.0 min
[3] The routes are calculated based on the same categorization of risk levels: *full-safe levels*={L0, L1, L2}, *partial-safe levels*={L3, L4, L5}, *non-safe levels* = $\emptyset$

TABLE II

CALCULATED RESULTS CONSIDERING DIFFERENT CATEGORIZATION OF RISK LEVELS

Vehicle ID	r_{max} (min)	Partial-safe range	Total Risk (min)	Distance (km)	Total waiting time (min)	Arrival time (min)
v_1	6.0	$[0.1, 1]$	5.9	7.6	0.0	17.1
v_2	6.0	$[0.05, 0.3)$	0.4	8.7	0.6	19.3
v_3	6.0	$[0.02, 0.05)$	0.0	9.5	0.0	19.4

Notes:
[1] Vehicles v_1, v_2 and v_3 have the same source and destination, and move at the same maximum speed of 30 km/h
[2] The routes are calculated by the algorithm, MOAAstar/Uncertainty, given r_{max} = 6 min and a departure time t = 0.0 min

and the same categorization of risk levels. We suppose that the responders in different vehicles have different amounts of oxygen, and thus each responder has its time constraint (indicated by r_{max}). Figure 14 shows a comparison of the results calculated by our system considering different r_{max}. As we can see from the figure, although the emergency task in this scenario requires all responders to go to the same destination, the system, which performs the calculation with the given r_{max} of each vehicle, generates different routes for them. The calculated results are shown in Table I. As shown in the table, while vehicle v_1 has to wait for a certain amount of time to avoid the moving obstacles, vehicles v_2 and v_3 can move without waiting, which makes them reach the destination earlier. The table also shows that our developed algorithm is capable of generating routes that have the total risk constrained by the user-specified r_{max}. With a higher r_{max}, the responders are allowed to stay longer in the obstacles, and thus follow a faster route to reach the destination. The scenario also shows that, following different routes, the responders would have different amounts of risk accumulated along their routes. If r_{max} is not considered in the routing, the total risk of the route that the responders could confront could be larger than their acceptable value, which would slow or even endanger their response process.

B. Scenario 2: Navigation for Responders With Different Preferences

Depending on the available protective equipment, responders may have different preferences. Therefore different classifications of risk levels would be made by them and used in the routing process. In this scenario, we compare the relief routes calculated based on the different categorization of risk levels (i.e., full-safe, partial-safe, and non-safe levels). The considered responders move at the same maximum speed, 30 km/h, and have the same r_{max} = 6 min. Figure 15 depicts 3 routes calculated for 3 vehicles. These routes have the same pair of source and destination, but are generated based on different categorization of risk levels. Table II shows the results of the calculated routes. As shown in the table, the total risk of routes are below the given constraint. The route generated for vehicle v_1 has the highest risk value, but it is still acceptable for v_1 and allows it to reach the destination in the least amount of travel time among the three vehicles. This is because that v_1 accepts higher risk-levels (as shown in Figure 15), which makes it possible to pass through the obstacles that are non-safe for the other vehicles. Waiting option is used by vehicle v_2 to avoid the non-safe obstacles and to reduce the risks. With our navigation system, the responders can get customized routes, using their own classification of risk levels based on their available protective equipment.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we provided an approach to support path planning in the presence of the uncertain moving obstacles, based on our earlier work with certain prediction of moving objects in [22]. We extended the previously developed data model and introduced the concept *risk level* to describe the status of road segments and junctions affected by the uncertain moving obstacles. The Monte Carlo simulation method is applied to quantify the influence of the uncertain obstacles on the road network. To generate the routes through the uncertain moving obstacles, we designed and developed an A*-based routing algorithm, MOAAstar/Uncertainty, which can handle the information of uncertain moving obstacles and incorporate the user profile into the routing. The application results demonstrate the capability of our approach in generation of routes through the uncertain moving obstacles.

It should be mentioned that, although in this paper we mainly consider uncertain boundaries of obstacles, our approach can also deal with the obstacles that have not only uncertain boundaries but also different properties (e.g., different concentrations of plumes, water depth). In the case of toxic plumes, the *risk level* can be defined based on the combination of the concentration and risk probability interval. For instance, *risk level* $= L1$ corresponds to $\{20, [0, 0.1]\}$, where 20 is the maximum concentration value of plumes, i.e., at most 20 milligrams per cubic meter (mg/m^3), $[0, 0.1]$ is the risk probability interval. With some extensions, our approach can also be applied to other types of disasters, such as fires and floods.

In the future work, we would like to perform the complexity analysis of the algorithm. By analyzing the algorithms, we can estimate how much memory and computation overhead would be required for the navigation system. This would help choose the hardware that is capable of running the algorithms

and quickly generating results. Furthermore, sensitive analysis of the algorithms will also be needed. To make our navigation system practically used, our approach will also need to be verified with responders. Interviews and discussions with responders would be necessary to help quantify the risk probability ranges to reflect risk in real disaster situations.

REFERENCES

[1] P. K. Chitumalla, D. Harris, B. Thuraisingham, and L. Khan, "Emergency response applications: Dynamic plume modeling and real-time routing," *IEEE Internet Comput.*, vol. 12, no. 1, pp. 38–44, Jan./Feb. 2008.

[2] E. Forcael, V. González, F. Orozco, S. Vargas, A. Pantoja, and P. Moscoso, "Ant colony optimization model for tsunamis evacuation routes," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 29, no. 10, pp. 723–737, Nov. 2014.

[3] J. C. García-Díaz and J. M. Gozalvez-Zafrilla, "Uncertainty and sensitive analysis of environmental model for risk assessments: An industrial case study," *Rel. Eng. Syst. Safety*, vol. 107, pp. 16–22, Nov. 2012.

[4] M. Glemser and U. Klein, "Hybrid modelling and analysis of uncertain data," in *Proc. 19th Congress Int. Soc. Photogram. Remote Sens. (ISPRS)*, Amsterdam, The Netherlands, 2000, pp. 491–498.

[5] J. P. Gonzalez, A. Dornbush, and M. Likhachev, "Using state dominance for path planning in dynamic environments with moving obstacles," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2012, pp. 4009–4015.

[6] J. P. Gonzalez and A. Stentz, "Planning with uncertainty in position an optimal and efficient planner," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Aug. 2005, pp. 2435–2442.

[7] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.

[8] Y. Liu, M. Hatayama, and N. Okada, "Development of an adaptive evacuation route algorithm under flood disaster," *Annu. Disaster Prevention Res. Inst., Kyoto Univ.*, vol. 49B, pp. 189–195, 2006.

[9] D. Mioc, F. Anton, and G. Liang, "On-line street network analysis for flood evacuation planning," in *Remote Sensing and GIS Technologies for Monitoring and Prediction of Disasters*. Berlin, Germany: Springer, 2008, pp. 219–242.

[10] V. Narayanan, M. Phillips, and M. Likhachev, "Anytime safe interval path planning for dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2012, pp. 4708–4715.

[11] B. Neuman and A. Stentz, "Anytime policy planning in large dynamic environments with interactive uncertainty," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2012, pp. 2670–2677.

[12] B. Neumany and M. Likhachevy, "Planning with approximate preferences and its application to disambiguating human intentions in navigation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2013, pp. 415–422.

[13] K.-C. Ok, S. Ansari, B. Gallagher, W. Sica, F. Dellaert, and M. Stilman, "Path planning with uncertainty: Voronoi uncertainty fields," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2013, pp. 4596–4601.

[14] M. Phillips and M. Likhachev, "SIPP: Safe interval path planning for dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Shanghai, China, May 2011, pp. 5628–5635.

[15] A. Rahman and A. K. Mahmood, "Feasible route determination using ant colony optimization in evacuation planning," in *Proc. 5th Student Conf. Res. Develop. (SCOReD)*, Dec. 2007, pp. 1–6.

[16] S. Sonti, N. Virani, D. K. Jha, K. Mukherjee, and A. Ray, "Language measure-theoretic path planning in the presence of dynamic obstacles," in *Proc. Amer. Control Conf. (ACC)*, Jun. 2013, pp. 5110–5115.

[17] J. Steenbruggen, P. Nijkamp, and M. van der Vlist, "Urban traffic incident management in a digital society: An actor–network approach in information technology use in urban Europe," *Technol. Forecast. Social Change*, vol. 89, pp. 245–261, Nov. 2014.

[18] J. P. V. D. Berg and M. H. Overmars, "Roadmap-based motion planning in dynamic environments," *IEEE Trans. Robot.*, vol. 21, no. 5, pp. 885–897, Oct. 2005.

[19] I. Visser, "Route determination in disaster areas," M.S. thesis, Dept. GIMA, Utrecht Univ., Utrecht, The Netherlands, 2009.

[20] Z. Wang and S. Zlatanova, "Taxonomy of navigation for first responders," in *Progress in Location-Based Services*, J. M. Krisp, Ed. Berlin, Germany: Springer, 2013, pp. 297–315.

[21] Z. Wang and S. Zlatanova, "Multi-agent infrastructure assisting navigation for first responders," in *Proc. 6th ACM SIGSPATIAL Int. Workshop Comput. Transp. Sci. (IWCTS)*, New York, NY, USA, Nov. 2013, pp. 1–6.

[22] Z. Wang, S. Zlatanova, A. Moreno, P. van Oosterom, and C. Toro, "A data model for route planning in the case of forest fires," *Comput. Geosci.*, vol. 68, pp. 1–10, Jul. 2014.

[23] H. Xue, F. Gu, and X. Hu, "Data assimilation using sequential Monte Carlo methods in wildfire spread simulation," *ACM Trans. Model. Comput. Simul.*, vol. 22, no. 4, Nov. 2012, Art. no. 23.

Zhiyong Wang received the master's degree from Tongji University, China, in 2010 and the Ph.D. degree from Delft University of Technology, The Netherlands, in 2015. His research interests include disaster management, intelligent algorithms, spatial data modeling, and analysis.

Sisi Zlatanova is an Associate Professor with the Department of Urbanism, Faculty of Architecture and the Built Environment, Delft University of Technology, The Netherlands. Her interests include 3-D modeling, management, and analysis of data.

Peter van Oosterom received the M.Sc. degree in technical computer science from Delft University of Technology in 1985 and the Ph.D. degree from Leiden University, The Netherlands, in 1990. Since 2000, he has been a Professor with Delft University of Technology and the Head of the GIS Technology Section.