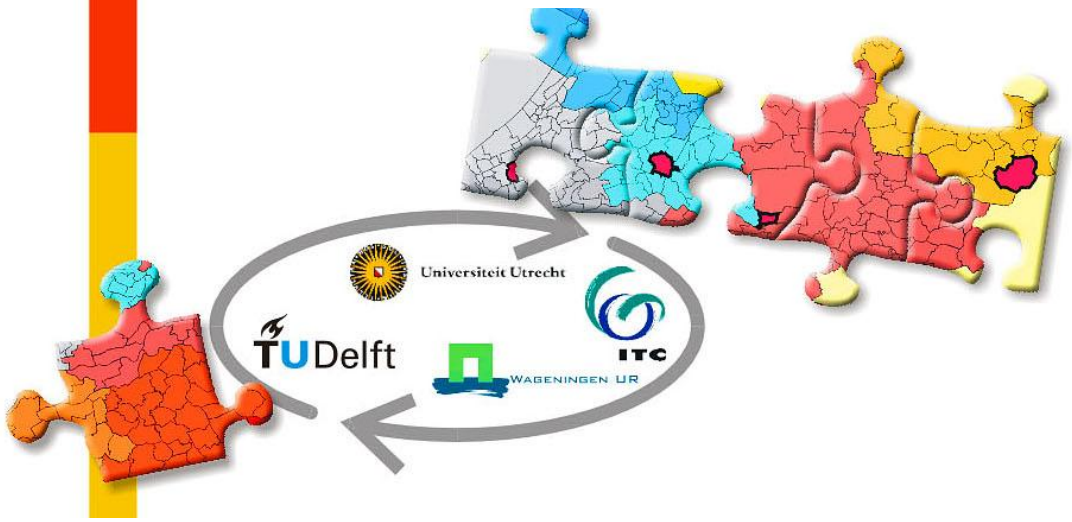
GIMA

Geographical Information Management and Applications

Route determination in disaster areas

Using predictions and introducing the option to wait to improve routing results

Ivo Visser



Route determination in disaster areas

Using predictions and introducing the option to wait to improve routing results

Author: Ivo Visser

Professor: Prof. Dr. Ir. P.J.M. van Oosterom
Supervisor: Drs. C.W. Quak
Reviewer: Drs. B.J. Köbben

Date: 28 August 2009

Abstract

Disasters caused by human action or nature are part of life. Preventing disasters from occurring is often not possible. Lives can be saved or lost depending on our response to the disaster. Determining the fastest route to the people in need is therefore important. The determination of this fastest route is however not straight forward, since the environment the route is determined in has become chaotic as a result of the disaster. In this research attention focuses on ways to take the circumstances in the environment into consideration in an automated way. A distinction is made between changes that can be predicted and changes that cannot be predicted. Based on literature the latter turns out to be done best by choosing a process of re-evaluating the route determined. Based on the degree of dynamism a static, semi-static, iterative or dynamic approach can be taken to this re-evaluation. In a static approach the route is determined only once based on the most current information at that time. When the circumstances change re-evaluation of that determined route can take place every time a change occurs (semi-static), every couple of minutes (iterative) or constantly (dynamic). In this research a preference was given to use a combination between the semi-static and the iterative approach. This results in a re-evaluation after a number of changes have occurred and if there are only little changes the re-evaluation should take place after a set time. The incorporation of predictions in the algorithm is a second focus area. By incorporating predictions one is able to anticipate changes to the network and take them into consideration in the route determination process. Incorporating predictions on plume movement and bridge openings and closings introduces also a need to balance between travelling extra kilometres and waiting. To investigate the implications of incorporating predictions into the routing process a routing algorithm was designed. The Dijkstra algorithm was adapted to read the closing times from a file and decide whether it is better to wait or take an alternative route. Tests show that the estimation of travel times are more accurate when these are created with the algorithm that incorporates predictions into the routing process. Based on this research it is concluded that using the adapted algorithm routes can be determined that will prove to be faster and safer than the result of a shortest path calculation based only on the travel costs in the network.

Acknowledgement

A lot of people played an important role in the creation and completion of this thesis. I would like to take advantage of this opportunity to thank them.

First of all I would like to thank Wilko Quak and Peter van Oosterom for their valuable and much appreciated feedback. Also I would like to thank them for brainstorming with me on possible directions for the research to take.

I also would like to thank Sisi Zlatanova for helping me pick a research subject from all the interesting options and opportunities that were presented.

My gratitude goes to Barend Köbben who volunteered to be the reviewer, when I needed one on short notice.

Furthermore my thanks go to ESRI Nederland, who supported and enabled me to finish my master. Special thanks goes to Jeroen van Winden for providing me with the means to perform my research.

I'm also grateful for the expert help I received from my colleagues at ESRI Nederland. Special thanks goes to Erik de Ruiter who helped me with implementing the algorithm in python. Also special thanks goes to Gineke Snoeren and Shuman Kibria for their expert input.

Apart from colleagues from ESRI Nederland, Robert Kieboom from CityGIS was very helpful in sharing his expert knowledge. Especially at the start of the research the information was very helpful to get a clear overview of the conditions in and around a disaster area.

Last but not least my special thanks goes out to my family and friends, who have been a great source of support throughout the process of completing this thesis.

Groningen, August 2009

Table of Contents

List of figures

List of tables

1 Introduction

1.1 Problem Description

The occurrence of disasters is part of life. They can be caused by either human action or by nature. Though disasters cannot always be prevented from happening, a lot of lives can be saved or lost depending on our response to the disaster. Losses can be minimized by quickly responding to a disaster through evacuating people from the disaster area, fighting the disaster and providing (medical) help to the victims. To do this a lot of organizations have to cooperate in a relative chaotic environment. This chaotic environment is caused by the scale of the disaster, the panic of the people, the destruction of roads, the scale of the help operations, etc. There is a risk in such a situation that the evacuation of people over available roads makes it hard for other organizations to get into the disaster area to fight the disaster. The disaster area might be contaminated with dangerous substances or the area might be flooded limiting the safe options of emergency response units to do their work. This information needs to be shared between the emergency response organizations, to improve the effectiveness of the operations and avoid unnecessary risks. As Geurts (2008) and Bruynooge (2008) state a central role is needed in such situations to deliver information to the different organizations.

The road network for example can suddenly change due to the disaster and route planners therefore need to retrieve the information dynamically from the information entered by the emergency response units to be up to date. Also information about the changing plume of gas is needed to make sure that the emergency response units are not guided right through the plume by their route planners. This research therefore focuses on the automatic determination of routes into the disaster area taking the changing gas plume and temporary closed roads into consideration.

To determine the shortest route to the disaster area the road network alone does not deliver sufficient information. In case of a disaster some roads might be damaged or blocked, they can be impassable because they are flooded, concealed by (toxic) smoke or in use as an evacuation route. The route generated should take all these factors into consideration to deliver the safest fast route to the disaster area. The most challenging issue in this case is that not only the position of the vehicles change over time, but also the road network and its accessibility.

Furthermore the concentration of a hazard in a particular area has implications for the ability to travel through that area. In case of a flooding an area covered by a 20 centimetres thick layer of water could be traversed if necessary whereas an area submerged for more than a meter cannot be traversed by car for example. The same distinction can be made for areas with different concentrations of toxins, though the ability to travel through that area is in this case related to health and not the physical condition of the network. Though it is not impossible to travel through the plume, the time one can remain inside the plume is restricted by the concentration of toxins. For the gas plumes another important restriction on the route to follow should be taken into account. This is the direction of the wind. If the wind blows the gas towards the emergency services the risk they end up inside the gas plume is considerably larger than when the wind is blowing the gas plume away from the emergency services. The route should therefore be calculated in a way that the emergency services approach the disaster area from the north when the wind is blowing from the north. This is called a

downwind approach and is advocated by NVBR(2006). The wind direction is also subject to change over time and so the time element remains an important part of the route calculations. Though wind direction is an important variable, the main aim is to take the changing gas plume into consideration. This gas plume gets its shape from the wind direction and speed. In addition the relative position of the plume with respect to its source is a direct result of the wind direction. This means that by taking the plume into consideration an important part of the wind characteristics are taken into consideration as well.

Position related to the Geospatial Data Infrastructure for Disaster Management project
This research will be conducted looking at the results of the RGI project: Geospatial Data Infrastructure for Disaster Management (GDI4DM). This project aims to create tools to respond better to emergency situations. These tools therefore should use the most recent information and deliver consistent results. In addition new information should be handled well by these tools, while data is coming from different sources both in the field and stored at the separate emergency services. The goal is to help the people working in the process of emergency response to base decisions on the most recent information in an easy way. To help them focus on actually fighting the emergency (GDI4DM, 2006). The determination and re-evaluation of routes is one of the goals. The people travelling should not have to be concerned with finding the route themselves. Apart from the inconvenience of having to navigate in a stressful situation, some invisible dangers, such as plumes of gas, can be incorporated by a computer where a person might not be able to spot them.
Though GDI4DM aimed to organize and use the most recent data, it does not necessarily aim to re-evaluate actions while these are performed, such as changing the route while driving to a destination. This is where the research deviates from the requirements as were set in this project.

1.2 Research questions

The research question is formulated as follows:

How can routes into and out of a disaster area be generated automatically taking a changing gas plume and waiting times into consideration.

The answer to this question can be found by answering the following sub-questions:

- Which methods are already used to use predictions in route determination
- Which dynamic elements influence the determination of routes
- How can a downwind approach be implemented within the route determination process
- Which events can change the availability of the road network during the disaster
- How can the changes to the road network be implemented
- In what way should the costs of travel for the disaster area be altered
- How can predictions of future changes be anticipated by the algorithm

- Under which conditions should a new route be calculated for a response unit already on its way

The determination of routes can be done either static or non-static. The static determination of routes would only use the available information that is current at the moment of determining the route. This approach thus becomes more and more unreliable with increasing route lengths especially in case of a disaster, where the road conditions can change in a split second. The re-evaluation of the route chosen therefore is very important, this corresponds to a non-static approach such as the semi-static, the iterative or the dynamic approach. By re-evaluating the route the determination of routes can be made more reliable, because an update in the information is incorporated in the re-evaluation of the route. Another approach would be to incorporate predictions into the route determination process. The reliability of such a route determination approach depends on the reliability of the predictions and hence the prediction models. Because these models for the prediction of congestion and the development of a toxic plume are not created here the incorporation of predictions into the route implies the creation of these models. This does however not mean that predictions done by external sources cannot be incorporated when determining the route. This method of determining a route is still static in a sense that the predictions done at the start of the process will be snapshots in time. Meaning that there is little interaction with the environment to alter the prediction made at the start based on the current situation once underway. In this research the determination of routes will be static in the sense that predictions done beforehand will be taken into consideration. This means that the route calculation is based on the information as is present at that moment and does not try to forecast for example the development of a plume or a change in wind direction. There are however assumed to be some externally provided forecasts of the plume location at different times, which will be incorporated in the route determination. Once these forecasts are altered during the trip, the route should be re-evaluated to take the new information into consideration.
The project will be built up in three phases. In the first phase a simple static route determination algorithm will be created and tested. In the second phase this algorithm is adapted to incorporate predictions. In the third phase the differences are examined and conclusions are drawn based on the findings from this phase. This comparison should help to draw conclusions on the actual benefits of determining routes using plume predictions for emergency services. In addition this separation into phases helps to split up the research in clear steps.

1.3 Methodology

The attention in this thesis is divided over two main areas. The first area is the re-evaluation of routes to adjust the route to comply with changes made to for example the network. This area is covered in literature by for example Montemanni (2002), Larsen (2001) and Golshani et al (1996). The incorporation of predictions is the second area attention will shift too. This will be done by implementing an adapted algorithm.

The first area of interest: Route re-evaluation
Apart from the more technical questions concerning the actual alteration of the travel costs, an important aspect in delivering a reliable route in a dynamic environment is the way to deal with changes occurring to the data. Montemanni (2002), Larsen (2001) and Golshani et al (1996) use route re-evaluation to deal with these changes. Two main options are using predefined re-evaluation moments or constant re-evaluation. In chapter two their approaches are described and classified.

Second area of interest: the incorporation of predictions into the route determination process
The main goal of this research is to establish a method to determine and re-evaluate routes using predictions of data that changes over time. Therefore a process will be developed and tested. Simulations will be used to test the performance in terms of outcome quality and changes to the process will result from these tests. The first phase will consist of determining routes in a static manner. The route should take road closings, wind direction and the disaster area into account. The second phase will focus on incorporating the temporal element into the analysis in the form of predictions. While the environment changes, for example when more roads are closed, the routes the emergency vehicles follow may have to be re-evaluated. In addition some roads that were closed might be open again by the time the routed subject arrives at that road segment. A third phase will focus on a comparison of the static routes and the routes that result from incorporating the predictions. The different phases are described in more detail below.

Phase 1: static route determination
During this phase only the data that applies to the moment of departure will be used for the initial determination of the route. The main focus here is on altering the costs of travelling along a road in a way that guides the emergency services around the plume and makes them travel down wind through the plume. This typically is a static case, where no predictions whatsoever are incorporated.
In a normal situation a route will be determined as shown in figure 1. This is what the commercial navigation software basically does. It calculates the shortest route to the destination. Now in case of a disaster the first thing that can change in the road network is that a road becomes blocked. In figure 2 such a road block is depicted. In this case a re-evaluation of the route is not necessary because the road block is not located on the determined route. In figure 3 another situation is depicted. Here the road block is located on the route determined initially and a re-evaluation of the route is necessary. This will result in an alternative route. In both these cases it doesn't matter whether the re-evaluation of the route is done continuously or only in case a road block proves to be on the current route. In cases where a block is removed however a re-evaluation of the route might result in a faster route. So the re-evaluation of the route cannot be limited to changes occurring on the current route. Another thing that is important to incorporate is the gas plume. It is important to take into account that a gas plume cannot be determined without accepting some uncertainty. This means that one is most likely already in danger when getting close to the plume drawn on a map. This means that determining a route that stays out of the gas plume alone is not a satisfactory solution. The route determined should also keep some distance from the plume even if this means that some extra time is needed to get to the disaster area when doing so. The exact distance that should be kept however is not clear. The idea is depicted in figure 4 and figure 5.
The assumption that the uncertainty around the outline of the plume is already included in the plume provided to the algorithm is made during this phase. The effects of this assumption are described during this phase as well.

When this phase is completed the route can be determined taking wind direction and the gas plume into consideration. It does however not incorporate any predictions for the gas plume. Therefore the result of this phase will both be input to phase 2 and phase 3. Phase 2 will consist of incorporating predictions into the determination of the route. During the third phase the results of both phases will be compared.

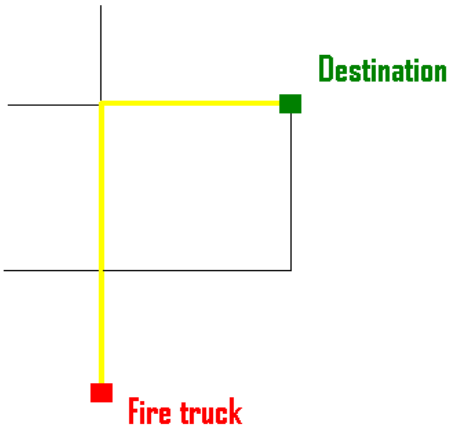


Figure 1: Normal route

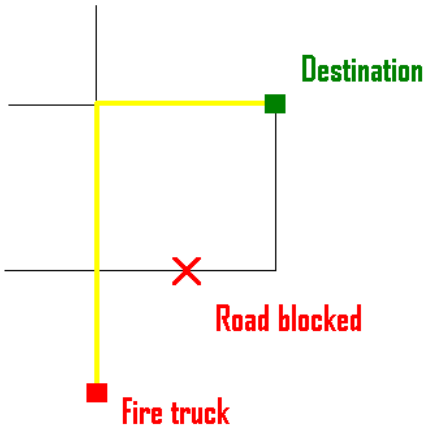


Figure 2: No re-evaluation necessary because the block is off-route

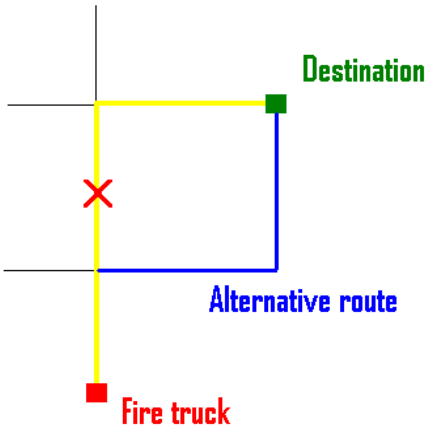


Figure 3: On-route Block leading to re-evaluation

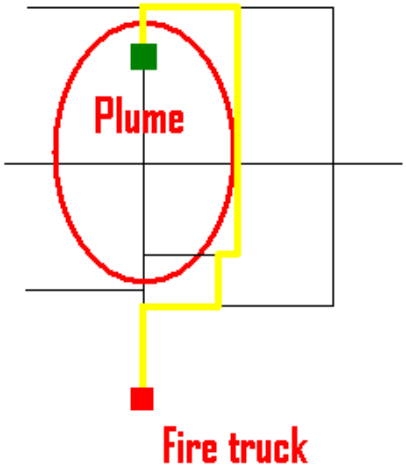


Figure 4: Routing around the plume

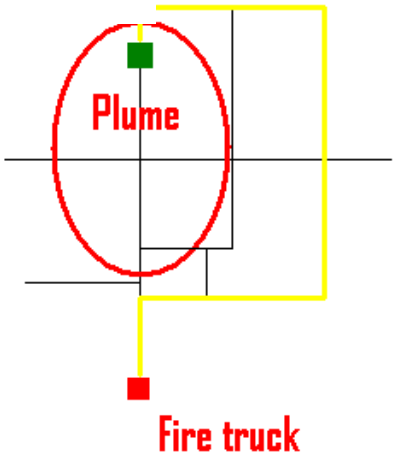


Figure 5: Routing around the plume taking a security zone into consideration

Phase 2: incorporating predictions into the route calculation

Though responding properly to the challenges presented by the environment helps to create a safer and maybe faster route to the destination, another improvement is expected to be made by incorporating predictions into the process. The focus in this phase is on incorporating the prediction of road closings and openings for bridges. Another prediction that can be incorporated is the prediction of the changing plume. Because predictions are always concerned with a measure of uncertainty the main focus here is not on the prediction itself, but on the incorporation of these predictions.

In relation to the gas plume predictions, three different cases can be thought off. These three cases are summed below.

1. Predicted plume does not interfere with the determined route: in this case the determined route does not have to be altered
2. Predicted plume spreads over the determined route: the determined route needs to be altered in a way that no conflict arises with the plume in interval T1 – T2 (see figure 6)
3. Predicted plume drifts away from the determined route: whether the determined route should be altered depends on the time it takes the fire truck to get to the edge of the plume. When the fire truck arrives at the edge of the plume before T2, waiting at

approximately the blue dot might be the fastest and safest option. When the fire truck arrives at the blue dot after T2 the fire truck can without a problem continue its way to the green destination point (see figure 7).

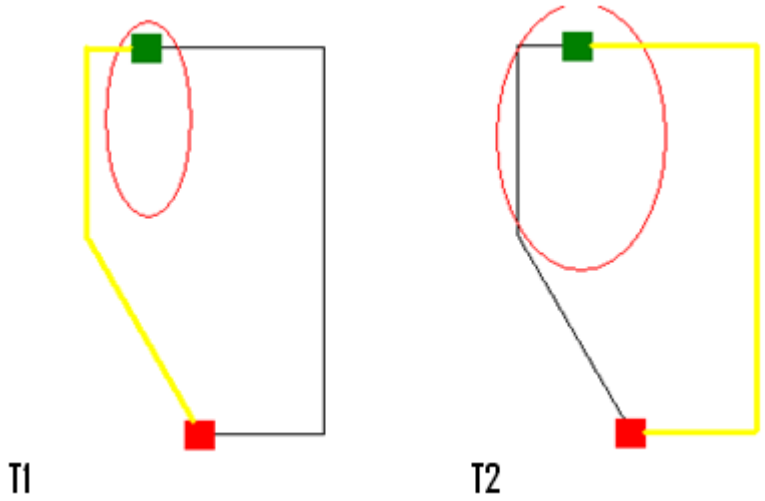


Figure 6: Alternative route as a result of the plume prediction

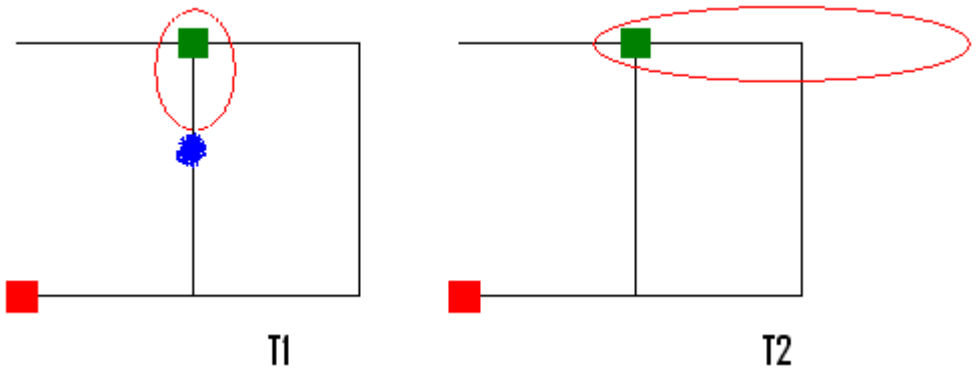


Figure 7: Possibility to wait for the plume to move away from the route

Incorporating these three possible actions asks for the creation of a method to use time dependent travel costs. These should not only be added up but even compared with each other across time and space. The possible outcome might be to either wait for the plume to pass, drive around the plume or the conclusion that the plume will already be gone when the fire truck arrives. This decision will be made based on predictions that will be incorporated into the algorithm created in this phase. This will be done by extending the Dijkstra algorithm. To investigate the implications of this adapted algorithm for the routes, the algorithm is tested for five scenario's.

Phase 3: Comparison between the simple static routes and the routes respecting predictions

Though in theory the usage of predictions in routes can have rather big advantages, the question whether it has advantages in practice remains unanswered. Therefore a comparison between the result of the first and the second phase will be made here. This comparison will be done by creating a number of scenarios for which both a static route and a route

incorporating predictions will be determined. Based on the test results conclusions can be drawn on the value of including predictions in the process of route determination.

Focus of the research

Because the different response units (fire fighters, police, ambulance) have different data needs, the focus in this research will not be on all of these units. For this research the focus will be on the fire fighters, because this response unit has to deal with a lot of information relating to navigation. The fire fighters are the people that go into the disaster area to fight it and collect information about toxics. Apart from information collected by themselves they also receive information about road closings, evacuation routes and the location of a field hospital from the other units. The method used to determine the routes should however not be that different for the other response units.

Apart from determining the target group also a choice needs to be made for the aspect of route determination to focus on. The focus can either be on the differences in possibilities and performance between for example a GIS and a database. Another aspect can be to determine which algorithm for the computation of routes suits this case best. These two elements focus on a comparison between different methods or approaches and not as much on the actual determination of the route in relatively chaotic conditions. Two elements that do relate to the chaotic conditions present during a disaster are the time element and the network element. The time element consists of the incorporation of predictions into the determination of routes and the incorporation of unexpected changes. Examples of these predictions can be a moving plume of smoke, a moving flooded area or bridges that close and open. The network element focuses more on the determination of the suitable subset of the complete network to use in a specific case. This means that some kind of classification method should be developed to automatically determine which road network should be used depending on the severity of the disaster and the type of response unit. In this research the focus will be on incorporating the wind direction into the route determination and secondly the incorporation of the time element. The main reason for this is that the challenge in a disaster setting is to determine a relatively safe route that is as short as possible.

Scope of the research

Though routes are determined from a particular point towards a particular destination, this research does not aim to deliver a route planner. Apart from the route line no user interface, spoken directions or address locators will be created.

Apart from roads being closed because of the disaster, some disaster response units might be able to leave the paved roads. For these units the determination of routes should also take the dirt roads into consideration. For the other units the severity of the disaster can determine whether the cycle paths for example will or won't be used. This means that the availability of roads not only results from the damage caused by the disaster but also by the type of vehicles used and the severity of the disaster. Though this would improve the results of the route determination process, this does not involve another way of determining routes. Merely this means that the input network is different for the various types of vehicles and disaster levels. Though the determination of travel times or costs for cycle paths and dirt roads can become a rather challenging task it will not be considered in this research. Another challenging task would be to allow the emergency units to cross open areas, such as fields in a park or rural area. Bemmelen et al (1993) wrote on this subject comparing the results of different algorithms. Though leaving the road could improve the results and help to evade the plume it is not considered in this research.

1.4 Structure of the thesis

All information needed to answer the research question is structured in a number of chapters. In chapter two attention will turn to literature on approaches to incorporate changes in the information into the routing process. In the third chapter possible changes in information in a disaster area are explored to arrive at some assumptions and needs for the design of an algorithm. The assumptions that have an influence on the adaptation of the algorithm as well as the user needs related to the route determination in disaster areas can be found in chapter four. The fifth chapter focuses on the design of the algorithm and the presentation of a couple of test cases. The first basic routing algorithm is created and tested in chapter six. In chapter seven the routing algorithm created in chapter six is adapted to incorporate predictions. In this chapter a description is provided transformation of the static Dijkstra algorithm into a non-static algorithm that is, an algorithm based on the Dijkstra algorithm that incorporates predictions into the route determination process. The adapted algorithm is also tested on the cases in this chapter. Based on the results of the algorithms used in the sixth and seventh chapter a comparison is made in chapter eight. In chapter nine the results are discussed. Conclusions based on the results of this research are drawn in chapter ten.

2. The process of routing: dealing with changing information

2.1 Introduction

In this chapter approaches to accommodate for changing circumstances when determining a route are presented. These approaches are collected from literature and present ways to re-evaluate a route that was created using a static algorithm. The first paragraph introduces the subject. In the second paragraph attention focuses on the routing process as a whole and in the third paragraph the approaches to the re-evaluation are classified and described. Apart from that the importance of using predictions is signalled. To get an overview of ways to incorporate these predictions some routing algorithms are described in the fourth paragraph.

2.2 General requisites for routing

In every route calculation information is needed on the origin, the destination and a network. The origin is the position where the route starts and the destination is the position where the route ends. The route is calculated between the origin and a destination based on a network. A network can be a set of lines or an area. The choice between the two depends on the type of transport a route is calculated for and the type of network. Lines are most used to represent road or cable networks and areas can be used to represent open sea, open land or the sky. Though using lines or areas seems an important difference in determining a route, the actual process runs down to determining the costs per element or distance. In the case of lines the costs can be determined per line segment. In the case of areas a cost can be calculated per distance unit within a cell for example (Bemmelen et al, 1993). The actual calculation of the route consists of searching for the route with the lowest costs, though algorithms for line networks differ from raster algorithms. A more challenging task is to deliver a route that is best in qualitative terms. Calculating the safest route for example not only challenges one to quantify safety but in addition asks for a balancing act between the length of the route and the safety of that route. Once these costs are determined they can either be considered constant or changing over time. Apart from the costs, the destination(s) and the network can be dynamic too.

The aim of the routing process is to minimize the cost of travel. The circumstances and the network the optimization takes place in is in many cases different. The main focus here will be on some researches that focussed on incorporating dynamic data in the process of routing (for Example: Min et al (2006), Montemanni et al. (2002), Larsen (2001), Golshani et al. (1996), Wang and Crowcroft (1992), Wang and Crowcroft (1990)). This can either be changing costs of travel or a changing set of destinations. Though the two seem rather different, the process of dealing with this dynamic data shows similarities. Before looking into the different methods used to incorporate dynamic data, first some attention will be given to the process of routing and some basics of routing algorithms.

2.3 The routing process

Determining the optimal route is not something that is limited to only one type of network. This process is useful for all kinds of traffic along any kind of infrastructure. Golshani et al (1996) use an area as the medium to travel through, focussing their attention on a balancing act between safety of travel and travel distance using incomplete information. Wang and Crowcroft (1992; 1990) take a computer network as the medium for travel. Their attention focuses on dealing with a heavy traffic load and a dynamic network environment. Their goal

is to minimize congestion on the fastest route without causing congestion on the alternative route. Hanshar & Ombuki-Berman (2007) and Larsen (2001) on the other hand focus on the dynamic vehicle routing problem (DVRP), where destinations can be added while travelling. The aim here again is to find the optimal route between all the nodes that need visiting. Though the networks and the type of traffic can vary widely, the parameters that are input to the calculation are very similar. Di Caro et al (2008) state that routing problems often consist of a source and destination between which a shortest path needs to be established. Also in the cases above there is a travel cost associated with each segment of the infrastructure and there is a source and a destination. In all cases the travel costs for all segments in the network should be known or estimated for the result of the process to be reliable.
The algorithms found differ because these are designed to take a specific characteristic into consideration such as incomplete information on costs on a raster network (Golshani et al., 1996), minimizing congestion on a line network (Wang & Crowcroft, 1992; 1990) and the incorporation of additional destinations (Hanshar & Ombuki-Berman, 2007; Larsen, 2001).

Wang and Crowcroft (1990) distinguishes between four procedures in a distributed routing algorithm. Though these procedures apply to a computer network with a number of routers, that maintain a table with the shortest paths to all destinations in the network, it can be translated into four procedures that apply to navigating on a road network.

1. distance measurement: acquisition of data about travel times and availability of the road segments (current and predicted)
2. information distribution: delivering the information gathered and the changes made to this data to the algorithms or devices that are going to calculate a route.
3. route computation: based on the information delivered in the previous phases a route can be calculated and existing routes can be updated
4. packet forwarding: driving to the next point of the route

In contrast to computer networks these four procedures cannot be integrated within one algorithm. The first procedure consists of integrating data from numerous sources. For road networks one can think of traffic announcements and mobility data measured as well as predictions on road availability and plume movement.
For road networks Min et al (2006) advocate the use of previously measured travel times as the costs associated with a segment instead of calculating the costs based on road length and speed limit. The second phase is concerned with exchanging the information. The exchange of information can be done beforehand in cases where the information is considered constant. In the case of Wang and Crowcroft (1990) this assumption was made implicitly, since they state that the information gathered will be used to route that package to the next hop. A route consists of a series of hops with their intermediate network. The costs of an edge between two hops are considered constant for the time the routed package travel over this edge. For routing over networks that do not have these hops but do have dynamic information this phase is reinitialised for each re-evaluation. Larsen (2001) for example expects to receive information on travel times and the destinations. If the new destinations are added these are delivered by the second procedure and the route taking the new destinations into consideration is calculated in the third phase. In cases presented by Min et al (2006) and Wang and Crowcroft (1990) the route is calculated similarly as in the case of Larsen(2001), but information on travel costs gathered during the travel is reported back and used as input for the routing of new subjects. The actual route computation is done in the third phase. The information gathered in the second phase is entered in an algorithm that is often tailor made to incorporate the type of data gathered (Examples are: Larsen, 2001; Golshani, 1996; Wang and Crowcroft, 1990).

The final procedure is to guide the package or the vehicle, along the optimal route determined, to its destination. During this procedure no measurements are made on the actual travel time in case the travel costs are considered constant. This is the case DVRP described by Larsen (2001). In cases where the travel costs are considered to be subject to change the first two procedures run within the fourth procedure. Golshani et al (1996) consider a case where a helicopter receives information on threats in its environment and adapts its own route based on this information. The information is partially retrieved from the environment by the sensors of the helicopter and partially from other sources. Relating this back to the four procedures identified one can see that the routed subject in this case acquires information, receives information from external sources, calculates it route, and travels along the route autonomously. During the fourth procedure information is gathered and the route recalculated and the course adapted if necessary. In the research of Min et al (2006) the integration of the procedures differs from the case presented by Golshani et al (1996) in that the result of the subjects information gathering does not mean that the subjects route is re-evaluated. The information is gathered during the travel but delivered to a database. The database can be queried again to route a new subject. In this case the four procedures are used and the first two procedures are also incorporated into the fourth procedure, but the information gathered during the travel is delivered to a database and does not become input for the subjects own third phase.
Based on the use of the different procedures in the different researches a difference between the travelling subjects between vehicles and packages becomes apparent. Packages in computer networks as described by Wang and Crowcroft (1990) have no control over the route they take to their destination. The hobs in the network decide in which direction the package is going to travel next. Vehicles are on the other hand in control of their travel. These vehicles have the information on their travel on board and could re-evaluate the route based on new information (case of Larsen (2001)), in addition the vehicle can gather and deliver travel information to a database (case of Min et al (2006)) and ultimately the vehicle could re-evaluate the route based on information just gathered (case of Golshani et al (1996)). Based on this it can be stated that if the travelling subject is a vehicle it fulfils the role a hob has in a computer network to some extent.
Differences in the capabilities of the travelling subjects determine the extent to which it is possible to find reliable routes in different environments. If the travelling subject is unable to gather information from the environment itself it has to rely on information gathered and delivered by others. In environments where network attributes are rapidly changing this might mean that the information in this case reaches the travelling subject too late. In very dynamic environment travelling subjects that gather their own information and share that information with other travelling subjects are more likely to find the actual optimal route than travelling subjects that do not gather and share information.
For the incorporation of the new information into the route the existing route should be re-evaluated. All the new information is entered again into the third procedure; the procedure where the route is determined.

2.4 Handling data changes in the routing process

When it comes to iterating the route determination process, there are a couple of approaches that can be chosen based on the situation the route determination takes place in. Based on Montemanni et al. (2002), Larsen (2001), Golshani et al. (1996) and Wang and Crowcroft (1992) the following approaches can be distinguished:

1. Static route determination: a route is calculated using the data available. The route is not corrected or recalculated when changes to the original data occur.

2. Semi-static route determination: this approach expands the static approach by recalculating the route every time a change occurs to the data.
3. Iterative route determination: a route is calculated using the data that is available. The route is recalculated at the end of a time window. This time window can for example be 1, 2, 3, 4 or 5 minutes. The route is recalculated every time whether there have been any changes or not.
4. Dynamic route determination: the calculation of the route is performed constantly. This means that the calculation is restarted as soon as it is finished.

The first approach to routing is the simplest approach. When choosing this approach one assumes that the network is not changing or only changing on the long term (Wang and Crowcroft, 1992). According to this statement this approach can be expected to deliver the best result in stable environments. Since changes in travel costs are not considered, the shortest route found does not have to be the shortest route in practice. The result of the static route determination approach is therefore sufficient for finding a route between an origin and a destination. If one however aims to arrive at the destination as fast as possible, finding the actual fastest route becomes more important. In this case one wants to have more information about the actual travel costs on the segments between origin and destination. The travel costs can be influenced by congestion for example. Because this information is only valid for a relative short period, the travel costs of the route may change during the travel. To take these changes to the travel costs into consideration a semi-static route determination approach can be adopted. Using this approach a change to the travel costs leads to a re-evaluation of the route. This means that the information used to calculate the route is as up-to-date as possible. Though the travel times are updated, routing choices made just before the update limit the options to find a faster alternative. The difficulty finding a faster alternative are likely to be larger when one is getting closer to the destination.
Larsen (2001) and Golshani et al. (1996) state that calculating a series of static situations is the easiest method but does not necessarily deliver the optimal solution. The alternative however is to accept a sub-optimal route at first, hoping a future event will better fit in with this route than it fits in with the optimal route. This involves accepting a lot of uncertainty and possibly accepting a sub-optimal route that remains sub-optimal throughout the entire trip. Larsen (2001) and Golshani et al. (1996) therefore choose to always search for the optimal route based on the information available at that time. When one is able to predict more and more events, the risk of an optimal route turning out to be sub-optimal as a result of an event becomes smaller. The incorporation of these predictions can be done by having time-dependent travel costs. Also Montemanni et al (2002) signals that even though this approach, re-evaluating routes without incorporating predictions, might not deliver the optimal result, choosing the optimal route based on the available information at that moment is the best way of dealing with uncertainty.
Larsen (2001) states that a disadvantage of this approach appears when the environment becomes more dynamic. Following this approach the re-evaluation of a route will be interrupted when changes are occurring in quick succession. Larsen (2001) proposes a slightly different approach to overcome this disadvantage, which is the iterative route determination approach.
In the iterative route determination approach the route is recalculated every couple of minutes. Though the third approach is expected to perform better in more dynamic environments than the second approach, the semi-static approach aims, in contrast with the iterative approach, to be up-to-date at every moment in time.
The fourth approach is designed to deal with rapidly changing circumstances. The continuous recalculation of the route will in practice be similar to the third approach. The recalculation

with the dynamic approach will have to deliver a result before being restarted. The time window of the successive calculations will in this approach be the same as the time it takes to determine a route.

When the degree of dynamism of one kind of environment changes over time, it will be harder to tie one specific approach to the particular kind of environment. This asks for a combination of approaches that is generic in a way that it uses up-to-date information without calculating the same route over and over again when the information has not changed. It is in such an environment that the dynamic route determination approach would seem the best approach at first. The downside would be that the route is still recalculated when the environment is in a relatively stable phase. The approach that best suites the relatively stable environment is the semi-static route determination approach. This approach however will have disadvantages when the environment becomes more dynamic as was pointed out earlier. During the more dynamic phase of the environment an approach such as the iterative approach will be more suitable. As a general approach to environments with a changing degree of dynamism a combination between a semi-static and an iterative approach seems most appropriate. This hybrid-approach should re-evaluate a route when a change has occurred during the relatively stable phase of the environment and switch approaches towards the iterative approach when clusters of changes are recorded during the more dynamic phase of the environment.

When looking at the actual re-evaluation process one can choose to re-evaluate the entire network or only the portion of the network that is affected by the changes. The latter of the two is more efficient in terms of calculation time and resources (Xiao et al, 2003). Xiao et al (2003) developed a method to limit the rebuilding of the network to the part of the network that is affected by the change. This means that the algorithm they developed checks whether a distance has increased or decreased. If there has only been an increase in travel cost on edge B, all child nodes including the end node of edge B are re-evaluated. A similar thing happens when there is a decrease in travel cost only on edge B and the total travel cost to the from node of edge B plus the new travel cost along edge B is smaller than the total travel time to the end node of edge B. In textbox 1 the method of this algorithm is clarified even more. Implicitly this algorithm is an implementation of the semi-static re-evaluation approach. Only a change to one of the edges is triggering a re-evaluation. This approach is according to Larsen (2001) cumbersome in a very dynamic environment. In a less dynamic environment however the benefits of this approach to partial network re-evaluation can prove to be considerable, because the route determination can be done quicker and the resulting route is always up-to-date. Important here is that the destination of the travel is not subject to change.

A combination between the approach advocated by Larsen (2001) and the approach used by Xiao et al (2003) would deliver an algorithm that re-evaluates at best once every 1,5 or 10 minutes. The re-evaluation will only take place if any changes occurred in the previous time window and the re-evaluation will only be done for the parts of the network that are affected by these changes. Though this approach seems to be a combination between two approaches that proved to be efficient under different circumstances, the combination of the two will not necessarily deliver the same benefits as the two separate approaches.

Approach to partial network re-evaluation (Xiao et al, 2003)

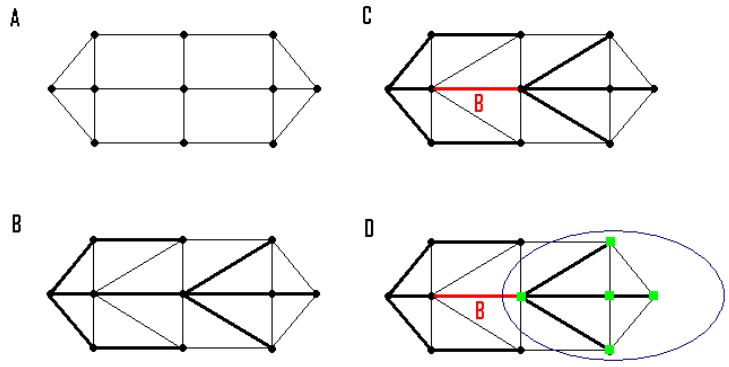


Figure 8: Partial network re-evaluation (edited from Xiao et al., 2003)

In the figure above a hypothetical network is depicted in frame A. For all the nodes in the network the shortest path can be determined. The bold edges in frame B show the shortest paths to the different nodes, in the pseudo code below these are depicted with the letter E. In frame C the travel cost of only edge B (in red) is changed. Edge B has a from node (BF) and a to node (BT).
In a simplified version of the algorithm the following happens to determine whether nodes need to be re-evaluated or not.

When the travel cost of edge B changes: TC(b) → TC'(b)
If TC'(b) > TC(b) **AND** part of E**:**
Re-evaluate route to BT and its descendants, taking all edges into consideration that have BT or a descendant of BT as its end node
Allowing for descendants of BT to be assigned to another ascendant
Else if TC'(b) < TC(b) **AND** costs BF + TC'(b) < costs BT**:**
Re-evaluate route to BT and all its descendants, taking all edges into consideration that have BT or a descendant of BT as its from node
Allowing for new descendants of BT to be found
Else:
Wait for other changes to occur

Textbox 1: Partial network re-evaluation

Although the approaches differ considerably, they have one important similarity. All of the above approaches use the most up-to-date data for the calculation, but no predictions of the travel times by the time the driver is assumed to be at that location. This means that the determination of the route is in all cases reactive. Consider the below example, where the travel times on one of the roads towards the destination city have changed considerably by the time the driver arrived at that segment. When this was known at the moment of departure, the at first longer alternative would have proved to be the faster route in the end.

Example: Travelling around rush hour
A driver is travelling between two cities. When mainly using highways there are two alternative routes. Route A is 5 km longer than route B. Route B however is well known for its traffic jams around rush hour. Now the navigation device finds route B, because this is the shorter route. By the time the driver is half way route B the rush hour has started. On route B the delay because of traffic jams is 1 hour more than the delay on route A. Looking back the driver would rather have chosen the longer alternative to bypass the traffic jams and arrive at the destination earlier.

Using predictions, in the form of time dependent travel costs, in the process of routing adds an approach to the approaches mentioned before. If all the changes relevant for determining the fastest route could be predicted the disadvantages of re-evaluating the route can be countered. Incorporating predictions in the algorithm will remove uncertainty and thus the disadvantage brought forward by Montemanni et al (2002) of previous routing choices preventing the optimal route to be delivered eventually. Similarly the risk of not re-evaluating the route using the semi-static approach, because a new change occurs before the previous change has been incorporated into the route, identified by Larsen (2001) will be countered as well. Based on this one can conclude that incorporating predictions into the route determination procedure will improve the resulting route.

The static, semi-static, iterative and dynamic route determination approach have in common that the information available when determining or re-evaluating the route are assumed to be effective immediately. In the case of a prediction part of the information available does not apply when the route is determined, but will become valid after a certain period of time. For the algorithm this implies that not only the identification of the next edge is needed, but also the time of arrival at that edge. Based on this time of arrival the right information contained in the predictions should be selected.

In the next section some routing algorithms and the way these deal with dynamic data are discussed. Even though these algorithms are reactive, these contain valuable insights in the way to deal with predictions. Since the predictions will be implemented by having time dependent travel costs it is important to insert the adaptation to the algorithm in the part where the travel cost of an edge is first retrieved from the data.

2.5 Routing Algorithms

For the calculation of routes there are a large number of algorithms available. Some of these algorithms will be explored here.

2.5.1 Dijkstra Algorithm

This algorithm assumes that there is a graph with a single source and all weights assigned to the edges of the graph are non-negative. It starts by updating the vertex closest to the source in a distance array (Puthuparampil, 2007; Cormen et al, 1992; Dijkstra, 1959).
The distance array contains all vertices at the start of the process. To indicate that none of the vertices have been processed, the distance values in the distance array are set to infinity for all the vertices at the start. What the Dijkstra algorithm does is merely updating the distance values in this array, starting with the source vertex which has a cost of zero (Puthuparampil, 2007; Cormen et al, 1992).

The values in this array represent the shortest distance found between the source and a vertex. The vertices that are now connected to the source through edges are added to a priority queue. A priority queue is a list of values that is ordered with the smallest value at the top. The minimum value in this queue is added to the distance array. This means that the shortest path to the vertex just updated in the distance array has been found. The vertex that was just updated in the distance array is removed from the priority queue. In the next round the vertices, that are connected through an edge with the last updated vertex in the distance array, are added to the priority queue. This process continues until the shortest path to all vertices has been found (Puthuparampil, 2007; Cormen et al, 1992; Dijkstra, 1959).
The Dijkstra algorithm implies a search in all directions. This means that in the case of a single destination that has to be visited, the search area is bigger than necessary. This disadvantage can be countered by implementing an algorithm that takes the approximate direction of the path to be found in consideration. An example of such an algorithm is the A* algorithm.

2.5.2 A* algorithm

The A* algorithm uses a heuristic estimate of the distance from any node to the destination (Puthuparampil, 2007; Hart et al., 1968).
The heuristic distance from any node to the destination is based on the straight line distance between that node and the destination (Lester, 2005). The way the algorithm searches for the shortest path from the source to the destination using this heuristic estimate is described below.

Add the starting point to the closed list. Next add all the traversable neighbours of the node in the closed list to the open list.
The distance to the nodes in the open list is a summation of the distance between the nodes and the heuristic distance from the node in the open list to the destination.
Add the node in the open list with the smallest total distance to the closed list, storing also the predecessor node. Again all the traversable neighbours of the nodes on the closed list are determined and the distances of these neighbours are (re)calculated. Again the node with the smallest total distance is added to the closed list. This process continues until the destination node is reached (Lester, 2005; Hart et al, 1968).
The difference with the Dijkstra algorithm is that A* does not search equally in every direction. This is done through defining a heuristic value that guides the searching into the right direction. The Dijkstra algorithm has no heuristic value so the search is done in all reachable directions. The downside of the approach of Dijkstra is that is takes more time to find the shortest path to the destination. If one wants to find the closest destination from a group of destinations the Dijkstra algorithm has the advantage over A* (Lester, 2005).

2.5.3 Bellman-Ford algorithm

The concept of this algorithm is similar to the Dijkstra algorithm. The main difference is that this algorithm is able to handle negative costs (Puthuparampil, 2007; Black, 2005). To make sure that a shortest path exists, the algorithm checks the graph for negative cycles (Black, 2005; Cormen et al., 1992). If a negative cycle exists in the graph, a shortest path cannot be found. This is caused by the fact that travelling along edge A with a negative cost will lower the total travel cost. If a route exists back to the start of edge A with a net negative weight the costs have become smaller when arriving back at the start of edge A. Travelling along edge A will again lower the costs. Hence there is a negative cycle where travelling along a set of

edges, connected to each other, will lower the total costs every time. If one is calculating the shortest path to any destination one will find that the shortest path will only be within reach when travelling an infinite number of times along the cycle. The destination will therefore never been reached. In case of routing over a road network, as is presented in this research, the costs are unlikely to be negative.
The algorithm updates the distance array for the source vertex and sets its value to zero. All adjacent vertices are relaxed and updated in the distance array. This procedure is repeated for all the edges. The second step is to visit all the vertices in the graph again to make sure that no negative cycle exists. A boolean value is returned indicating whether a solution exists. If there is no negative cycle the algorithm returns that a shortest path exists (Black, 2005; Cormen et al, 1992)

2.6 Towards an algorithm for routing in a disaster area

In the case of a DVRP a number of locations have to be visited, of which only a certain amount are known beforehand. This means that the destinations one has to travel to are dynamic. In this case Larsen (2001) states that the more information is known beforehand the more optimal the route can be determined. This statement advocates the usage of predictions in the route determination process to arrive at better results. To incorporate the predictions a routing algorithm should be adapted to be able to evaluate time-dependent travel costs, as was determined earlier.

Though in the DVRP the destinations are considered dynamic, those can be considered relatively constant in case of a disaster. The dynamic information is in the case of a disaster more likely to be a block that need to be avoided instead of destinations that need to be visited. The actual goal however remains the same; the best route should be determined.
The theory on the approaches to the re-evaluation apply to the case of a disaster since new information should be incorporated into the route similarly as in the cases described earlier. Larsen (2001) states that the emergency services operate in a very dynamic environment. Reasons he mentions for this are incomplete data, uncertainty about the needs of customers and their location. In general however the destination the emergency services are dispatched to are relatively clear. According to Kieboom (2008) the emergency services are gathered at field stations outside the disaster area and dispatched from there. The routing towards this field station does therefore not involve uncertainty about the destination. However in case of toxics the uncertainty concentrates around the spreading of the toxics. This spreading changes with time introducing a dynamic component into the determination of the routes. Larsen (2001) expects that including more a-priori information and expectations will improve the routes found.
By incorporating predictions on road availability, by defining time-dependent travel costs, into a routing algorithm part of the dynamic information becomes known beforehand. This reduces the uncertainty about road availability and thus reduces the degree of dynamism of the environment. Following this argumentation a choice could be made to reduce the re-evaluation approach from dynamic route determination to iterative or even a semi-static route determination approach. The need however to work with the most up-to-date information combined with a risk of still encountering a large number of changes in a very short period. The best approach of re-evaluation therefore is considered a combination between the semi-static and the iterative route determination approach.
Because ideas on re-evaluation of the routes have already been developed and implemented (for example: Montemanni et al., 2002; Larsen, 2001; Golshani et al., 1996; Wang and Crowcroft, 1992) attention will focus here on incorporating the predictions into an algorithm. Before developing and testing such an adaptation to an algorithm the dynamic information for

the case of a disaster, which is considered here, should be listed. This is done in the next chapter. Also in this chapter a concept of the adaptation of the algorithm is presented.

3. Data changes in case of a disaster

3.1 Introduction

The process of route determination consists of a couple of elements. These elements are information, route determination and user response. In order to get the optimal result all these elements should be integrated perfectly with each other. The different elements are described separately in this chapter.

3.2 Information element

The information element is build up out of two things:

1. the location information; the source and the destination of the route to be determined,
2. the network information; the actual travel costs (nodes and edges) and the available roads

In both groups the information can either be changing or static. For location information the source can be a vehicle driving around the network and the same applies to the destination. In many cases the destination is assumed to be a static location, such as an address, an intersection or a building. In case of a pursuit or an interception of a vehicle, the destination is a moving object and thus a form of changing information. Information on the velocity, the direction and goal of the moving destination helps to predict the location of the destination by the time one is able to be somewhat closer to that destination. If the prediction is of good quality one is able to determine the location where one can arrive or intercept the moving destination. The source can also be a dynamic location. For the route determination however the source is often assumed to be static during the determination. With network information all information on road availability and travel times are meant.

The road availability consists of a static and a changing component. The roads in the network represent in general the roads that are available to a vehicle to navigate on. Based on static restrictions or characteristics some of the roads can be unavailable to particular modes of traffic (bus, train, car, bicycle, foot, four-wheel-drives), closed in a particular direction (one way traffic and restricted turns) or impose restrictions on passing vehicles based on their dimensions (maximum height, weight, width or length). All this information is static and ideally known beforehand. It is however possible that due to some event the availability of the roads is increased or decreased. These events introduce a temporal component to the availability of the roads. For example some roads can be closed to particular types of traffic because of tough winds (trucks on dikes for example) or rubble alongside the road (closed for traffic with a particular width). On the other side roads can be closed to all types of traffic because of some form of danger or damage (for example: road construction, large accident, collapsed bridge, flooding, toxics, danger of explosion, etc.). All these events change the availability of the roads and therefore the static network. Another thing these events share is that they happen at a particular moment. Vehicles that are already past the event location by the time the event takes place will not have to take this event into consideration.

In addition to the road availability the travel costs along each road segment are important for the determination of the optimal route. Similar to the road availability the initial costs are static and can be based on a combination of the speed limit and the length of the road segment. Though this is an easy and rather quick method to estimate the travel costs of a road segment, it is rather optimistic as Min et al (2006) show in their research. The travel costs generated using this estimation method assume that one has a constant velocity along the entire road segment not taking into account the time needed to accelerate to that velocity and

the time needed to decelerate when arriving at the next intersection. In addition it takes additional time to make a turn or wait in front of a traffic light. This is an example of a cost at a node. Though these considerations apply to all roads in different degrees, the enhanced estimate of the travel costs taking all factors mentioned earlier into account is still an example of a static estimate. In practice however these travel costs will also depend on weather conditions, accidents, road maintenance, number of vehicles and the type of traffic (cars, busses or trucks). These are all examples of often hard to predict dynamic factors. Taking these factors into account could however improve the reliability of the route found and of the total cost estimate. Incorporating information on events like road construction, expected travel times and a link with the traffic lights system also turns up in work of the NVBR (2006). Within the network information one can also have temporal road closings or temporal changes in travel time.

All this information should be gathered and grouped in an efficient way for the algorithm to be able to access and use the information. The actual quality of the output depends on both the assumptions in and the quality of the algorithm and on the other side the quality and availability of the input information. The algorithm itself once created processes the information the same way every time. The differences in quality of the output between two cases is therefore most likely to be caused by differences in information quality.

The final and very important element is the response of the user to the route returned by the algorithm. This user response however is not limited to only the understanding of the meaning of the directions but also the ability to comply with the directions provided. If the user decides for any reason to leave the route provided to him by the route determination process, the total travel costs will change and the route will not be the most optimal one anymore. This sub-optimal result could have a higher cost than another sub-optimal result, that had a smaller chance of accidentally leaving that route.

Within the categories defined earlier some events that seem unlikely to appear can exist simultaneously in case of a disaster. For this particular situation it is therefore interesting to see how these events can be used in the route determination. First it is necessary to identify the kind of events that can exist in a disaster area that imply routing choices for the emergency units.

Location information

In case of a disaster the actual destination and source of the route can be considered static. The emergency vehicles are very likely to move towards a central gathering point close to the disaster area to be dispatched from there or perform their work at that location (Snoeren, 2009; Kieboom, 2008). For the emergency units that move into the disaster area to perform measurements the actual destination can be updated. The destination however is not a moving object but merely one or a series of static locations. The input to the process of routing can therefore simply be two points, a source and a destination.

Network information

The main dynamic dimension of the route determination in disaster areas is related to the information on travel times and road availability. In the case of a disaster a relative large amount of changes to the network can be thought of which are located in a relative small area. This means that the combined impact of the changes to the network can be relatively large. The correct implementation of this information can therefore help to considerably improve the quality of the route found as was advocated by Montemanni (2002), Larsen (2001) and Golshani et al (1996) in the previous chapter.

The availability of the network segments in a disaster area can be influenced by the damage to the roads and road blocks. Both these cases appear at some point in time on the particular

segment and are valid in many cases for a longer period. This can be anything from a couple of hours to a number of weeks or even months. For an emergency response unit it means that the period the road segment is closed is longer than the time it takes to travel to a destination. This information therefore typically asks for a re-evaluation of the route and not for the creation of models to predict the occurrence of damage or road blocks. Another thing that influences the road availability is the closing and opening of bridges and rail crossings. These are both temporal in that there state (open or closed) changes over time. The fact that the bridge is closed for traffic when the emergency response unit starts its journey doesn't mean that the emergency unit will have to wait for the bridge to go open again when it arrives at the bridge. The same goes for the opposite situation. If the bridge is open for traffic when the emergency response unit starts driving, that doesn't mean the bridge is still open when it arrives there. This typically calls for the incorporation of predictions for the open and closing times of bridges. Any delay during the travel to the disaster area can result in additional loss of life and commodities. Based on this idea it seems to be very interesting to avoid any bridges or rail crossings that might be closed at some point in time. This would no doubt result in a higher average speed during the travel. A route crossing a bridge, introducing two minutes of waiting time, might however still result in an overall shorter travel time.
The travel times themselves can also change. In general one can think of traffic jams and road construction. Around the disaster area an additional hazardous area might exist such as a flooded area or a toxic plume. When the layer of water is thin enough to travel through the travel costs can also be altered. As the characteristics of the flooded area change the travel costs of the flooded road segments should be adjusted accordingly.
In the cases considered here the destination is located inside the restricted area, but can also be located outside the restricted area. NVBR (2006) states that the restricted area should not be crossed. In cases where the destination is outside the restricted area, this is the desired solution for the routing problem. There can however be cases where the restricted areas need to be entered. The destination of the route is inside the area in that case. In these cases one simply has to enter the area in order to reach the destination. Since the NVBR (2006) advocates a downwind approach with respect to the restricted area, going into the restricted area will have to be done downwind as well. The possibility of a destination to be inside the restricted area means that even though the road segments in that area should in some cases be unavailable, these segments should be traversable if needed. The only way to keep them available in the cases that the destination is inside that area is by altering the travel costs. The alteration of the travel costs should be done in a way that it is highly unlikely that the route would lead through the restricted area when the destination is outside of it.

3.3 Route determination

Emergency units operate in a very dynamic environment (Larsen, 2001). This is also the case when a disaster strikes. A lot of unusual things can happen at the same time and ask for a quick and special response. In this situation a lot of information is gathered under a lot of pressure. The assisting systems such as a route planner should take some of the pressure away from the people fighting the disaster. All the information about the traffic situation and travel times can still be gathered the same way as would be done usually, but in a chaotic environment it is bound to change faster. When the information has been gathered, the information should be implemented in the route determination process. In addition temporal data are possibly present about closing and opening times of bridges and rail crossings. Apart from temporal blocks such as bridges, there can also be temporal blocks that remain for a longer period. Examples are road closings as a result of damage or artificial closings set up by the police. Because this information is likely to change once the route has been determined,

the route needs to be re-evaluated to incorporate the changing information. Temporal blocks temporary higher costs that exist on a road segments for some time, such as bridges, can be incorporated using predictions. Apart from the route calculation and the availability of the data a route can also change as a result of choices made by the user.

3.4 User response

Though the actual useful dynamic data for the algorithm is not found in the route determination and the user response element, both introduce a dynamic element into the process as a whole. In the case of the user response the actual actions of the driver form the dynamic information. As long as the actions of the driver comply with the route suggested no re-evaluation is necessary. Once the driver leaves the road by taking a wrong turn or making a U-turn, the current location of the vehicle is the input again for the re-evaluation.

3.5 Using predictions in the algorithm.

All these sources of dynamic information are to be considered once the incorporation of this information into a larger process is desired. In this case the algorithm is mainly focussed on the incorporation of the network information on road availability and travel times. In routing the complete set of road segments form a graph. Mostly the graph is considered to be 2D. With the incorporation of time dependent data into the route determination process a third dimension is introduced into the originally 2D graph. The third dimension is the time dimension in which the dynamic data and the implications of the time dependent data for the algorithm can be visualized.

A 2D graph consists of vertices and edges. Those are spread over a geographic area which will be referred to as the XY field in which the X is the west-east orientation and the Y axis represents the south-north orientation. All the edges connect to vertices and all the edges are part of the connected set of edges, the graph. When introducing the time element into the graph an extra axis is introduced, which will be referred to as the Z-axis. The location of the second vertex will be an X, Y, Z location that represents both the geographic location of the vertex as well as the minimal travel time to that point. In figure 9 the graph is shown in 2D, the Y-axis is left out of the picture since all the vertices have the same value on the Y-axis. Figure 10 depicts the 3D graph, as can be seen the points all have the same value on the Y-axis. The graph shown in figure 9 makes it easier to demonstrate the idea of the 3D graph and the incorporation of predictions.

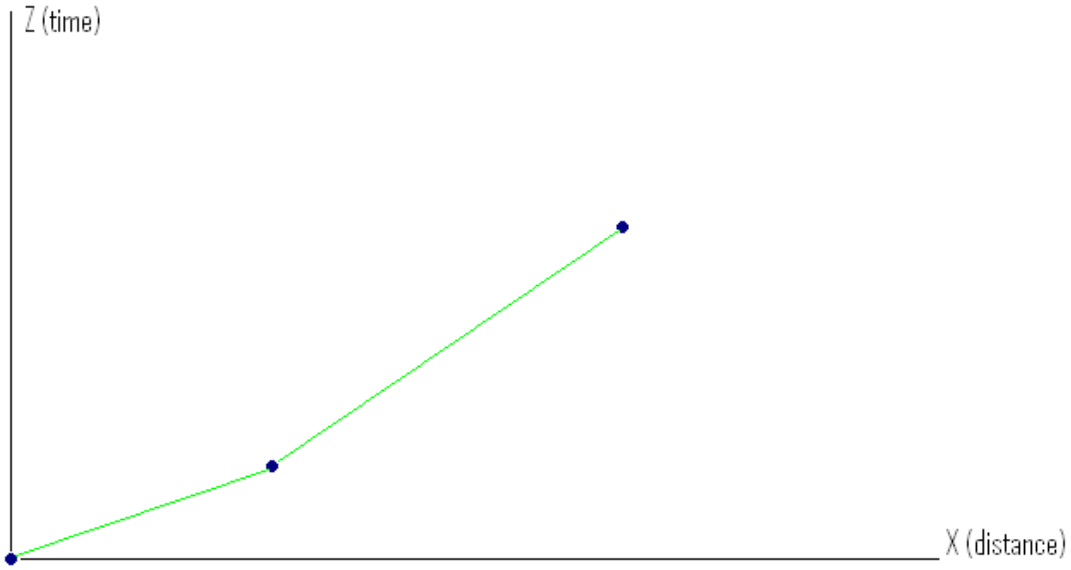


Figure 9: 2D graph

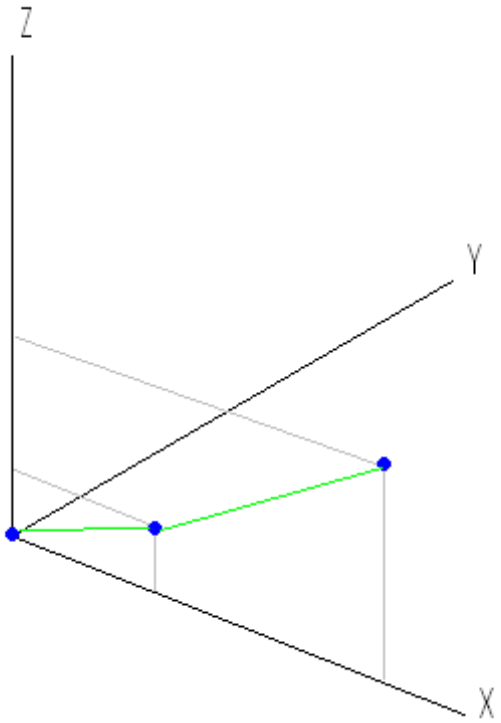


Figure 10: 3D graph

The travel time between two points depends on the distance between these points and the speed of travel. In figure 9 and 10 this is visualized by the slope of the edge connecting the two vertices. If the travel time or the travel cost increases the slope increases.
One of the dynamic components is the introduction of temporal blocks such as the bridges and the train crossings. This means that the possibility to wait is introduced to the graph which transforms all the vertices to lines spanning in the Z direction. All the edges that connect the vertices transform into polygons into the Z-direction. Someone travelling from vertex A to vertex B is able to move freely through the polygon connecting the two vertices. The result of this representation is that the unit can choose the speed of travel freely and is able to wait half

way along a segment. The unavailability of road segments because a bridge or rail crossing is closed limits the options of the unit to choose its own speed and waiting moments. These temporal blocks can be represented as a series of holes in the polygon. This is represented in figure 11. In this picture the block itself is represented as a straight line represented in black. The hole that is the result from this block is not entirely square but has a slope relating to the travel time along this road segment for the vehicles that move across the road segment just before and just after the block is in place. This drawing is therefore directional and only applies to situations where the vehicles are moving from A to B and not to situations where vehicles are moving from B to A.

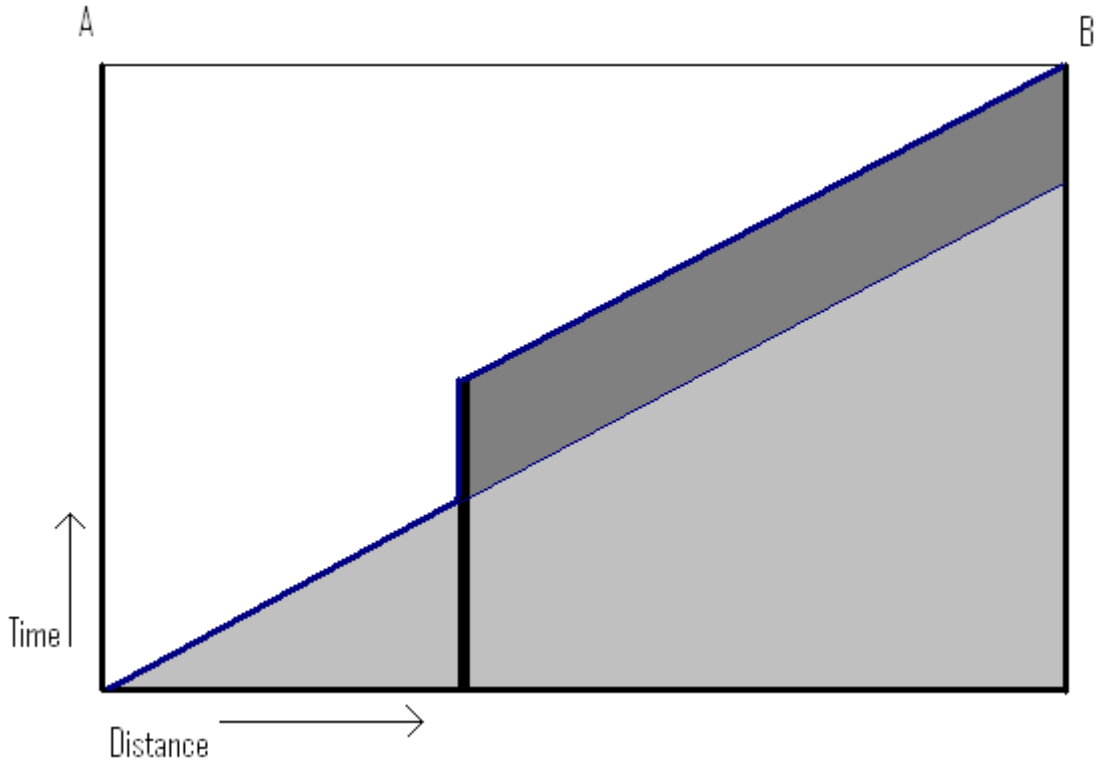


Figure 11: The influence of a Block on the 3D graph

The fastest route that can be found to a location is a route that results in the lowest value on the Z-axis possible for that location. Alternative routes can be identified as well. These are all the arrivals to the location with a higher value on the Z-axis. The difference in travel time can be determined using the difference between the values of the two arrivals on the Z-axis. In figure 12 an example is given of the way the information on alternatives can be derived from the 3D graph. In addition it is shown that in this case it is better to wait for a temporal block (green line) than to choose an alternative route (red line).

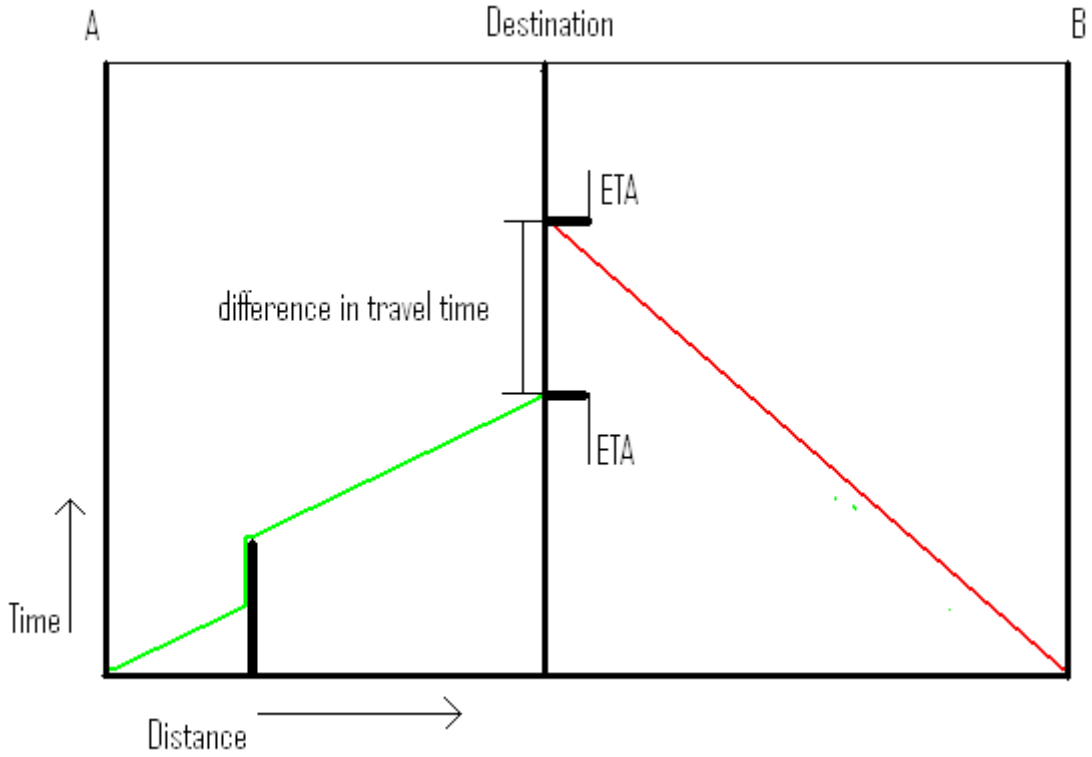


Figure 12: Getting information from the schematic representation of the graph

In this representation of the graph the time associated with crossing an intersection, making a left turn or making a right turn is not incorporated. These additional costs at a node will be referred to as the intersection impedance. This intersection impedance is not implemented in the current version of the algorithm. The implementation of this intersection impedance in the algorithm can be done in different ways, which can best be represented by modifying the 3D graph.

1. Introducing a waiting time, which is the same for each intersection that is crossed.
2. Introducing a waiting time that differs based on the turn one takes. For example a small impedance is encountered when turning right, a slightly higher impedance for moving straight ahead and the highest impedance for making a left turn.
3. Incorporate the average intersection impedance in the average travel time for a segment.
4. Changing the travel costs of the first and the last part of the road section similar to option 1
5. Changing the travel costs of the first and the last part of the road section taking the kind of turn that one is about to take similar to option 2

This graph only applies to situations where waiting is an option and not to situation where the speed of travel can be chosen freely. This is caused by the slope of this 3D graph. In cases where the speed can be chosen the slopes can take all possible forms. Though there is obviously a maximum speed that is possible on a particular road and with a particular car, but that speed can hardly be determined. This also means that if the temporal blocks can be placed half way along a road segment, that the maximum feasible speed should also be determined within the road segment. This leads to a rather complex situation in which the speeds driven can differ between road segments and within road segments. Within the road segments the difference depends on the road design, the widths and bends that are present.

4. Assumptions and user needs for the adapted algorithm

4.1 Introduction

Before the Dijkstra algorithm is described in chapter 5 and the improvement of the algorithm in chapter 6, the assumptions and user needs that have a bearing on the algorithm are described here. The assumptions are described first followed by the user needs in the second paragraph.

4.2 Assumptions

4.2.1 Introduction

When determining a route for disaster response units a large amount of data needs to be processed. This is to ensure that the outcome of the routing process is a route that is both safe and fast. To arrive at such a result input is needed from different experts and expert models. In addition the location of the unit that needs to be guided and a representation of the network they need to be guided over is needed. The creation and acquisition of this information involves the creation of models, the execution of computations and the acquisition of information in the field. In the process of routing this information is however only the starting point. The process of routing aims to combine this information from these data sources to create new information, the optimal route. So in this research the focus is on the routing process, taking the information that is available for granted. This means that the processing and aligning of the units gps-location, the determination of the closed area (gas plume or flooded area), the models to predict the movement of this closed area, the travel costs and the network availability will not be subject of study here. The subject of this study will be how to combine and incorporate this information to arrive at an optimal route. To be able to limit the research area to this subject a number of assumptions are made which are described below.

4.2.2 Study area

The area of study will be limited to one city outline or part of a city. This is to limit the computation time and to keep a clear overview of the possible alternatives. Having a number of cities the potential influence of the city design on the results of the routing algorithm can be identified.

4.2.3 The network dataset

The route will be determined using a network of roads. The road data provided in the TeleAtlas Multinet set is assumed to be complete. The travel times present in the attributes of the roads are assumed to be good estimations of the actual travel times encountered on these roads in reality.
For developing a method to incorporate waiting times, a moving plume and a downwind approach the use of turn restrictions and one way traffic are assumed not to be important. In addition the emergency response units are assumed to ignore these restrictions during a disaster, when they are close to the actual disaster area. The actual method implemented is not different for a network without one way traffic and restricted turns, since the implementation of these two restrictions comes down to filtering the number of options one has at every intersection. The challenge in this case is to introduce a method to overrule this filter when getting close to the actual disaster area.

4.2.4 The plume and other restricted area

The method developed in the research will among other things aim to route a vehicle around a predefined area. This area could be any area the vehicle should not be travelling through. One can think of flooded areas, toxic plumes, fog, dangerous neighbourhoods, combat areas, environmental zones in cities or national parks. Here the case focuses on a disaster area where fire fighters have to navigate to the disaster area. During this trip they should travel as little as possible through the restricted area. The restricted area can be clearly outlined or be nearly invisible in this case. When an area is flooded the approximate outline of the restricted area can be seen from the field. In case of toxics this area is completely unclear when walking in the field without any additional information. Because of the difficulty of identifying the outlines of the area in this case, the added value of information sources outlining the restricted area is obvious. In practice models are used to produce these outlines, the use of this valuable information for routing is however not in place. An example of a model that is used in the Netherlands is the GasMal model. TNO (2009) describes the process of delivering such a model that is intended to provide information on the area affected by a chemical incident. The outline produced by such a model is assumed to be present, usable and correct for this research. Under these assumptions the result of the models can be used directly as input to the routing process. Any changes, uncertainties or inaccuracies should therefore be incorporated into the process of creating the plume or by pre-processing the input to the routing process. The input to the process should in all cases be the area one cannot, or doesn't want to be in unless absolutely needed.
Whether one cannot be in an area or doesn't want to be in an area results in a particular multiplication factor for the travel times within that area. This multiplication factor can be different for different parts of the restricted area, but in this case is assumed to be constant across the entire area.

4.2.5 Movement of the plume

In most of the cases mentioned above the shape and position of the restricted area can change over time. This change should be incorporated in the process of routing. Since it is assumed that the predictions of the restricted area are delivered by other models and can be used as-is as the input for the routing process any change to the shape or location of the area should be delivered as input as well. It is assumed that both the restricted areas are polygons and have an attribute stating the time it applies to in some sort of start and end time. Furthermore it is assumed that the end time of the first polygon coincides with the start time of the second polygon. This means in practice that it is assumed that at any moment in time there is only one polygon applicable as the restricted area.

4.2.6 Waiting times

Another consideration when developing the routing algorithm is to incorporate the possibility to wait. This means that a vehicle could wait for a particular duration. This duration is based on the time of arrival and the point in time where the road is traversable again. A temporary block of the road limits one to travel from the start of a road section to the end of that road section. When the temporary block is gone one can travel again from the start to the end of that road section. The waiting time therefore alters the time needed to traverse a road section. The amount of time one needs to wait depends on the time of arrival. The exact time of arrival at the actual block depends on the length between the start of the road section and the

temporary road block. It is assumed that the road sections within an urban area are relatively short, due to the intersection density, and the waiting times relatively long compared to the travel time along a road section. This enables the assumption that the time of arrival at the road block can be considered equal to the time of arrival at the start of the particular road section.
Waiting times can be used in relation to the plume and in relation to bridges and railway crossings. With regard to the plume the waiting time can be estimated for roads that are within the first plume, but not within the second plume based on the end time of the first plume and the time of arrival. This waiting time should be taken into consideration before the emergency unit actually drives onto the respective road section. This means that the waiting time is calculated based on the time of arrival at the start of the road section in the plume. This calculated waiting time is used for the determination of the route, but in the field the waiting time should be applied to the road section preceding the road section within the plume and be recalculated to make sure that the unit remains outside the plume until the time is equal to the end time of the first plume.

4.2.7 Emergency unit location

In order to be able to determine the source of the route to be determined the vehicles location is needed. This information is input to the router and is considered given. It is assumed that the position of the vehicle is located exactly on a road segment that is connected to the network. The actual process of snapping the location to the road network to deal with inaccuracies of either the network or the gps-location is assumed to be done outside of this algorithm and is therefore not considered in the algorithm itself.

4.2.8 Navigation

When producing a navigation device a number of models, processes and techniques are integrated. One can think of determining the position, designing an user interface, spoken instructions, the determination of the route and the display of the maps and the route. Except for the determination of the route it is assumed that all the other models, processes and techniques are in place.

4.3 User needs

Though from a theoretic point of view it is interesting to see how the predictions of data can be incorporated, it is also very helpful to determine whether this possibility is also desired in practice. The user needs described here are all related to the process of routing and data handling and represent the needs identified for the fire departments. All the information on user needs is derived from a report of NVBR (2006) and an interview with Snoeren (2009). The needs are grouped here in a number of themes.

4.3.1 The network configuration

Based on the priority a vehicle is travelling with restrictions such as one way traffic, restricted turns and maximum speeds should be switched off or on. Though this need applies to all roads, the highways are some sort of an exception to this rule. It is allowed to travel in the wrong direction along a highway, but it is only in very special situations that it is actually done.

4.3.2 Real time data

Apart from the static network representing the roads and a travel time based on road length and maximum speed there is a need to incorporate real time data in these travel times. Some needs are indentified that ask for the incorporation of information related to road construction and the real time traffic situation (such as traffic jams).

4.3.3 Dynamic data

In addition to the real time data there is a need to incorporate dynamic data into the routing process. The dynamic data is different from the real time data, because real time data is data that is delivered at the moment it occurs. This is a change that is not predicted and it can therefore only be incorporated by re-evaluating the route determined. The only difference with static data is that it is not the same every time. Dynamic data focuses more on the incorporation of the knowledge, that the data changes over time, into the routing process. This dynamic data consists of predictions of events and changes over time. Needs identified for this theme are the incorporation of the predictions of the plume. The need however is not as important since the plume changes relatively slow. More attention is asked for the incorporation of the predictions of open bridges. When the times a bridge is open are known beforehand this data should be incorporated in the routing process (Snoeren, 2009). The NVBR (2006) however have a different approach to this challenge. They identified a need to warn the bridge personnel that an emergency vehicle is coming their way. The idea is that the bridge remains open for traffic until the emergency vehicle has passed. Though the approach to the same problem is rather different the incorporation of one of the needs does not necessarily mean that the other should be excluded.

4.3.4 Routing process

For the process of routing the needs are a little less directly formulated. It is for example unclear whether a route should be recalculated when an alternative route becomes faster. The need to have a route be calculated automatically in the emergency vehicle itself is clear. The desire to have a route that remains the same during a trip or is adjusted every time a change occurs is not explicitly stated. What can be expected is that another route becoming five minutes faster than the current route due to a change in travel costs would lead to a desire to change the route. The exact boundary however is not stated in the documents. If a route is changed for every gain in travel time or travel costs the time associated with changing to that new route. For example when one has to take a left turn in for example 20 meters where the old route was going straight ahead at that point. The additional costs of performing such an unexpected manoeuvre might be larger than the travel time gained by changing to that new route. A clear desire that is stated is that the route should never cross the gas plume and that the destination when it is within the plume should always be reached travelling downwind. Snoeren (2009) however states that depending on the kind and the concentration of the gas the route might lead through the plume if a large additional travel time is associated with the alternative. Another need is to have the Estimated Time of Arrival (ETA) at the disposal of the emergency unit.

5. Algorithm design and design of test cases

5.1 Introduction

The initial step towards the implementation of the dynamic data into the algorithm is the creation of a reference situation. This first phase of the creation of the algorithm delivers such a reference situation. This means that the results from the algorithm that incorporates the dynamic data will be compared with the results from the algorithm, for the static situation, created here. During this phase only the data that applies to the moment of departure will be used for the initial determination of the route. The main focus here is on altering the costs of travelling along a road in a way that guides the emergency services around the plume and makes them travel down wind. This typically is a static case, where no predictions whatsoever are incorporated. The algorithm will later be extended to incorporate the predictions of closings. The algorithm is created here from scratch to make sure that the comparison is only and entirely based on the incorporation of the predictions. To test the algorithm some subsets of the road set were made. In addition information on the plume is necessary as well as an origin and a destination.

5.2 Test data and algorithm design

Because the road set covers the entire country containing a couple of million records some subsets were created to demonstrate the algorithm. Each set only contains the road segments of one city or part of a city to limit the calculation time and to keep an overview of the possible alternatives that exist. The subsets are presented in five cases located in three different cities with a distinct layout. The first case is located in Rotterdam, the second case is located in Groningen and the third case is located in Delft. These three cities were chosen because the location of the potential blocks are known by the researcher. This means that the cases present real life block locations and their potential impact on the determination of a route. To make it possible to alter and reset the travel times of the road segments one field was added to the table containing a copy of the original travel times. This field is called "St_Travel" and can be found a couple of times within the algorithm.

The source and destination points used for the algorithm have only one attribute that is used to denote the unit number the source and destination apply to. It is assumed that each emergency unit can be identified using its unit ID. This number has no particular length or logic behind it during this research. The number can be used within the algorithm as long as it is an unique integer and the main reason it is used is to differentiate between the cases. The source and the destination are located in different sets. The unit ID should be unique both within the source and the destination set but identical between the sets. This results in a set of one destination and one source with the same unit ID.

The third element that is input for the algorithm is a fictive plume. This plume is an ellipse stored in a shapefile without any additional attributes attached to it. The only function of the plume is to select the roads that cross the area in order to alter the travel costs of the road segment. The algorithm does apart from that nothing with the plume data. In figure 13 an example situation is shown that demonstrates the situation before the algorithm starts. Because a plume has a bearing on only the travel costs it is left out of this figure.

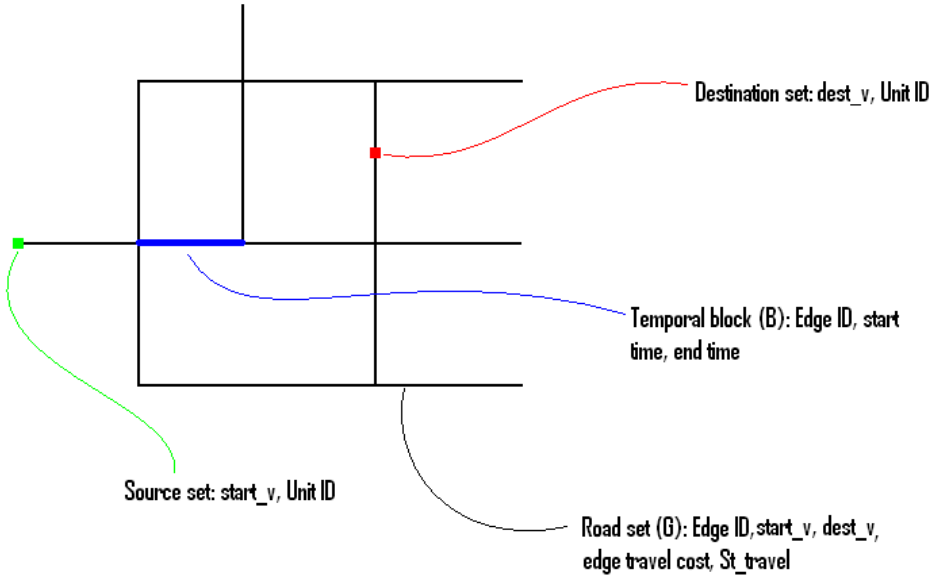


Figure 13: input information at algorithm start

Here it can be seen what information is stored in which set of data. First of all the source and the destination set is separated in two and they both have an unit ID attached to it. This unit ID could for example be 50, which means that the source is the current destination of unit 50 and the destination is the location unit 50 should move to. All the road segments are located in a road set depicted with the letter G. Here one can see that both the travel cost as the standard travel time (St_travel) for each edge is stored. St_travel contains the travel times along an edge and the travel costs of an edge also include a penalty for driving through a plume for example. The temporal blocks are collected in a different set and contribute to the travel costs of an edge. These additional cost represents the waiting time and is not stored with the edge, but added up during the computations. So the travel costs of an edge and of a route can be described with the following formula:

Travel costs = travel time + costs for driving through a plume + waiting time

Figure 14 depicts the initial step in which the edge the source is on (yellow) is added to a travel costs list with a travel cost of 0. This is to represent the fact that the unit is already on this edge, so there is no additional travel costs associated with traveling to this edge. The last node on this yellow edge is set as the start node of the iteration of which one is depicted in figure 15 and figure 16.

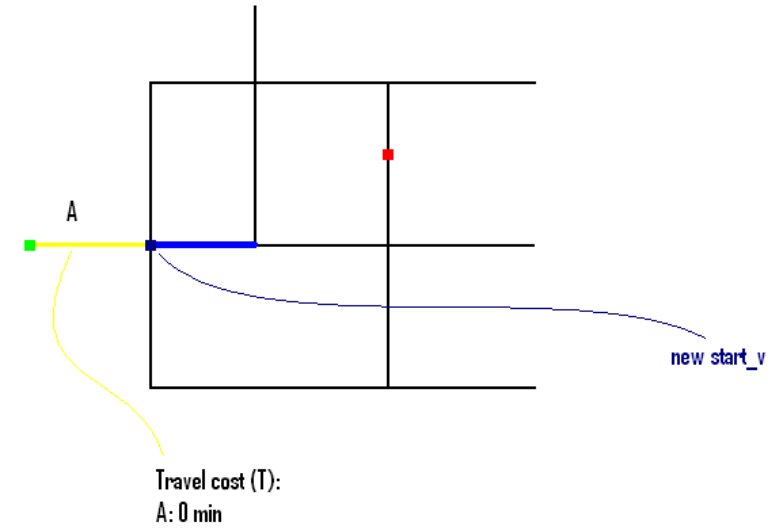


Figure 14: initial step of the algorithm

In figure 15 the three edges connected to the yellow start edge are collected and stored in a queue. Note that the travel time is corrected for the temporal block which is open at the time this edge will be reached.

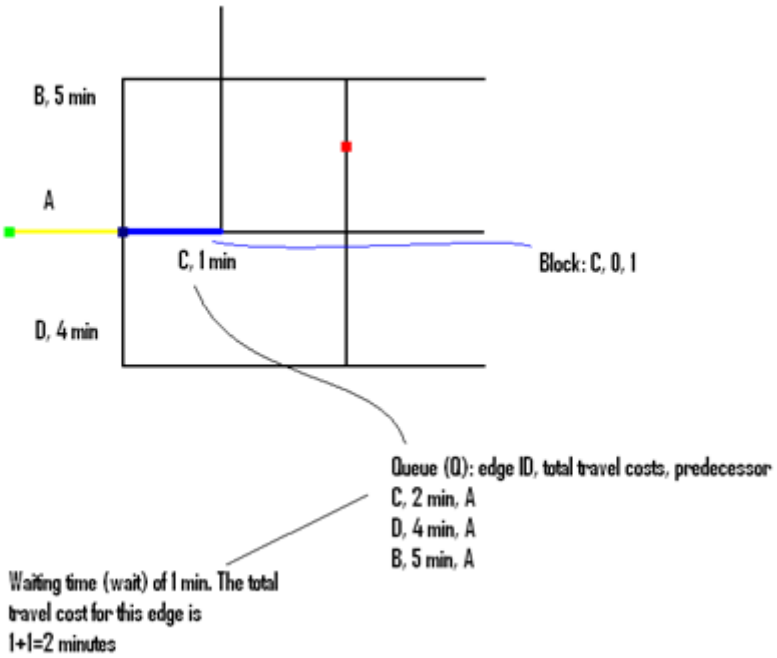


Figure 15: collecting the connected edges and storing them in a queue

In figure 16 the next step of selecting the edge with the smallest total travel cost from the queue is displayed. As can be seen here this step also involves adding this edge to the list with travel costs (T), storing the predecessor of this edge in a predecessor list (P) and deleting this edge from the queue.

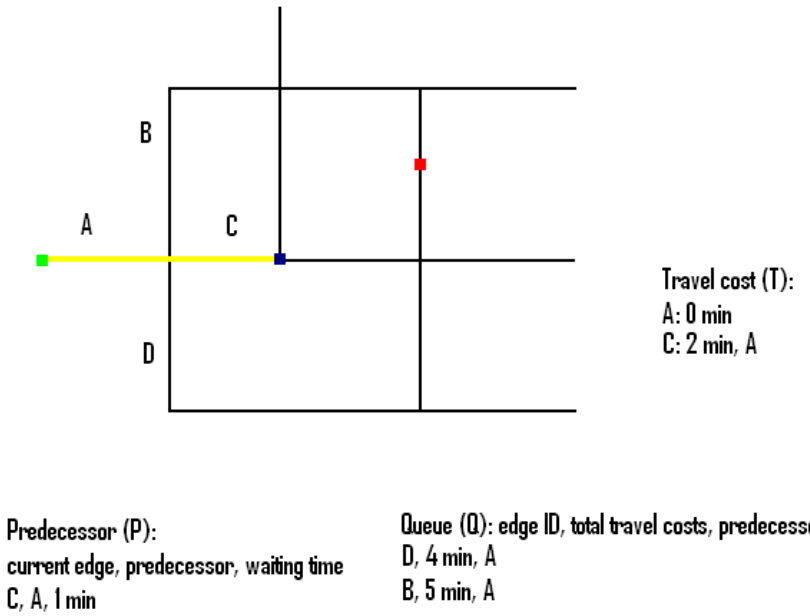


Figure 16: retrieving the edge with the smallest total costs

Edge C has now been explored and the blue node becomes the new start node in the search for connected edges. When the destination node has been reached by repeating the steps depicted in figure 15 and 16, the route (R) can be determined. This is depicted in figure 17.

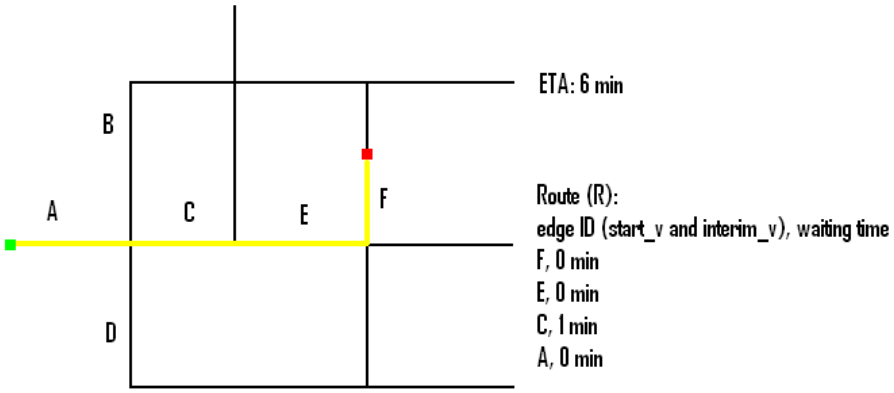


Figure 17: retrieving the route

This is done based on the predecessor list which helps to trace back the optimal route from destination to source.

In textbox 2 the pseudo-code of the algorithm to be created is depicted and the separate parts are linked to the figures.
Implementation issues and choices connected to the creation of this algorithm will be explained in chapter 6 and 7.

```
#################### Dictionaries and lists see figure 13 - 17 ########################
G       Graph
        Containing all the edges that participate in the network and their associated costs
        Values in the dictionary represent the from node, the to node and the edge costs
Q       Queue
        Containing all unexplored vertices that share an edge with a vertex that is already
        explored.
        Values in the dictionary represent the unexplored vertex, the predecessor of this vertex
        and the total travel cost from the starting vertex to this unexplored vertex.
T       Travel cost
        Containing the smallest travel cost to every explored vertex in the Graph including the
        starting vertex [1].
        Values in the dictionary represent the vertex and the total cost involved in arriving at that
        point.
P       Predecessor
        Containing every explored vertex with the predecessor of that vertex following the
        shortest path.
        Values in this dictionary represent all the explored vertices and the vertex number of
        their predecessor on the shortest path.
B       (temporal) Blocks
        Containing all the edges that are temporally blocked
        Values in this list represent the edges that have a temporal block on them.
R       Route
        Contains all the edges that are part of the fastest route.
        Values in this dictionary represent the edge identifiers (this could also be a layer in
        ArcMap)

################################ Variables: see figure 13 ########################
start_v = starting vertex[1]
interm_v = intermediate vertex
dest_v = destination vertex[2]
wait = waiting time
t = time
edge_tt = edge travel cost
tot_tt= total travel costs
edge_ID = edge identifier      # consists of start_v en interm_v in Graph
e = edge

################################ Initial: see figure 14 ##############################
start_v = origin
t = current time          # the start time of the trip
T[start_v] = t            # add the origin to the dictionary with the smallest travel times
tot_tt = 0
wait = 0
dest_v = destination
```

Textbox 2: pseudo-code for the algorithm

```
############################## Routing: see figure 15 ############################
While start_v not equal to dest_v:      # continue these steps until the destination is reached
        For start_v in G[edge_ID]:      # extract all edges where start_v is part of the edge
                                        # identifier
                interm_v =  edge_ID - start_v           # interm_v is the part of e that is not
                                                        # start_v
                tot_tt = T[start_v] + edge_tt           # tot_tt is supposed to be a clock time
                For interm_v - start_v or start_v - interm_v in B:      # if this line is in the set of
                                                                        # temporal blocks
                If tot_tt < B[end time] AND tot_tt >= B[start time]:    # and it is closed then
                        wait = B[end time] - tot_tt
                else:
                        wait = 0
                tot_tt = tot_tt + edge_tt + wait        # time of arrival at interm_v
                Append to Q[interm_v]: interm_v, tot_tt, start_v, wait          # add the other part
                                                                        # of the edge identifier to Q
##################### Change to a new current edge: see figure 16 ####################
        e = minimum(Q[interm_v])                # the least total travel time edge
        If interm_v not in T or tot_tt  < T[interm_v]:
                Append to T[start_v]: e (interm_v, tot_tt) # append the intermediate point and
                                        the total
                                        # travel cost from e to T
                Append to P[interm_v]: e (interm_v, start_v, wait)    # add the edge to the list of
                                                                      # predecessors
        delete e from Q[interm_v]               # make sure this edge is not used again
        start_v = interm_v                      # set the explored vertex as the current vertex

########################## Returning the route: see figure 17 ########################
If start_v is equal to dest_v:
        While start_v in P[interm_v]:           # as long as current vertex is a vertex with
                                                # predecessor
                Append to R: start_v, interm_v, wait            # add the line and not only the
                                                                # vertex to the route
                start_v = P[start_v]            # set the predecessor as the input of the next loop

While start_v in R[start_v]:                    # as long as current point is a from point in R
        select from G where edge_ID = R[start_v]-R[interm_v]
                or edge_ID = R[interm_v]-R[start_v]         #select line in G
        start_v = R[interm_v]                   # set the to point as the from point
        return total wait, selection in G       # return total waiting time and a selection that
                                                # represents the route
```

[1] The starting vertex is a location somewhere along an edge. The starting vertex can either be the vertex closest to the location of the actual starting point or both the from and the to vertex of the edge the starting point is located on.

[2] The destination is a location somewhere along an edge. The destination vertex can either be the vertex closest to the location of the actual destination or the first vertex reached on the edge the destination is located on.

Textbox 2: pseudo-code for the algorithm (continued)

5.3 Description of the cases

To be able to compare the results of the different phases, five test cases were set up. The test cases are located in different cities with different city designs. Apart from that the straight line distance between the source and the destination varies as well as the location and orientation of the restricted area. This was done to test the algorithm for different circumstances and trip lengths. The city design should be different between cases since it influences the number of fast alternatives and the difference in travel time between them. If the city is set up with square blocks and straight connecting roads the alternative route will involve a relatively smaller amount of additional travel time, than in cases where one has to move around the city centre on the other side because a road has become unavailable.

5.3.1 Case 1: Rotterdam

In this situation the source is located on an island. The destination is located in an neighborhood on the mainland. Between the source (green) and the destination (red) the plume is located blocking the route that at first sight seems to be the most straightforward one. The island in this case has a limited number of roads connecting the island to the mainland. It can be expected that waiting is an option here since the alternative road leaving the island connects to the mainland on the wrong side of the river with respect to the destination. The main roads in Rotterdam are relatively straightforward running through the center of the city. This could result in relatively straight routes. When such a main road is blocked however the alternative would include traveling towards another main road, since the other road segments are mainly positioned perpendicular to the main roads.

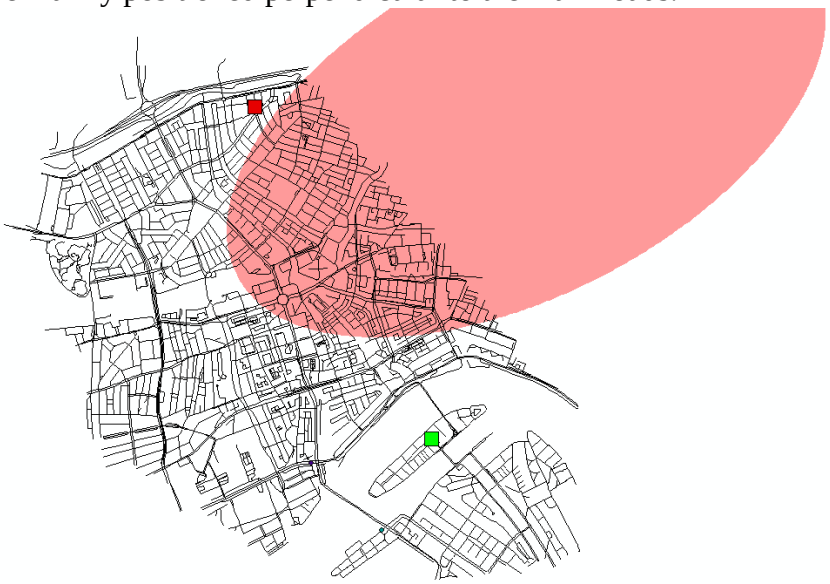


Figure 18: Case of Rotterdam

Reference route

Using the algorithm created in this research the reference route was determined. The plume input and the temporal blocks (bridges) were not included in this calculation. This route is the fastest route in terms of travel time based on the travel time attribute of the input set of edges. This reference route will be used for comparison with the results of the route determination process, taking the plume into consideration. One can see here that the blue line follows the main connecting roads for the majority of the trip.

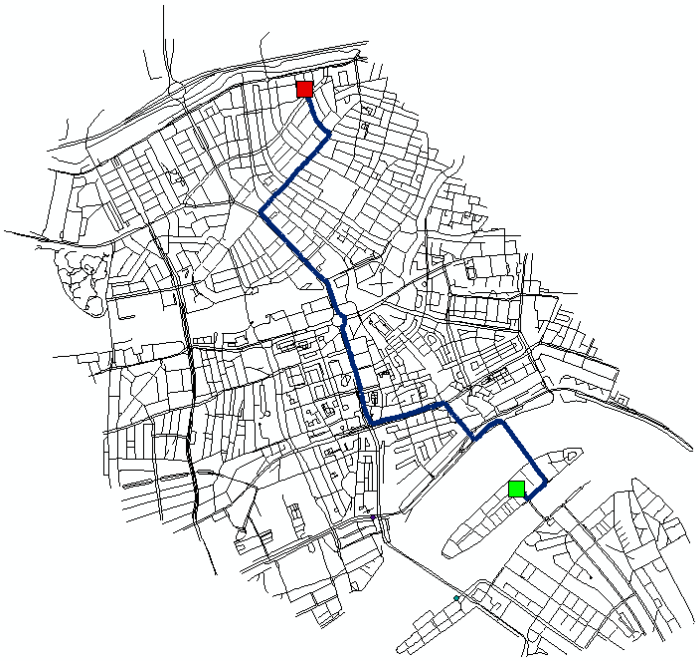


Figure 19: Reference route for the Rotterdam case

5.3.2 Case 2: Groningen

The second case is located in Groningen. The subset created here contains the roads in and around the city centre. The route to be determined most likely crosses the city centre connecting the destination (red) and the source (green) located on the edges of the subset. Between the source and the destination the plume is drawn. This plume is blocking the reference route in order to examine the impact of the plume on the route. This makes it possible to see the impact of the algorithm on the route more easily. The network in this case is characterized by ring roads positioned around the city center. Within these rings the area is split up is square blocks by the other roads. In contrast with the case of Rotterdam no major connecting roads are running completely from one side of the subset to the other side, crossing the city centre. The majority of the roads provide access to the main square (blue) in the city centre and are crossed by the ring roads. The city centre is surrounded by bridges. This means that when the bridge closing times are taken into consideration a balancing act between waiting for a bridge and entering the city centre over another bridge takes place. Here the plume blocks a considerable number of options, but still there is enough room for alternatives.

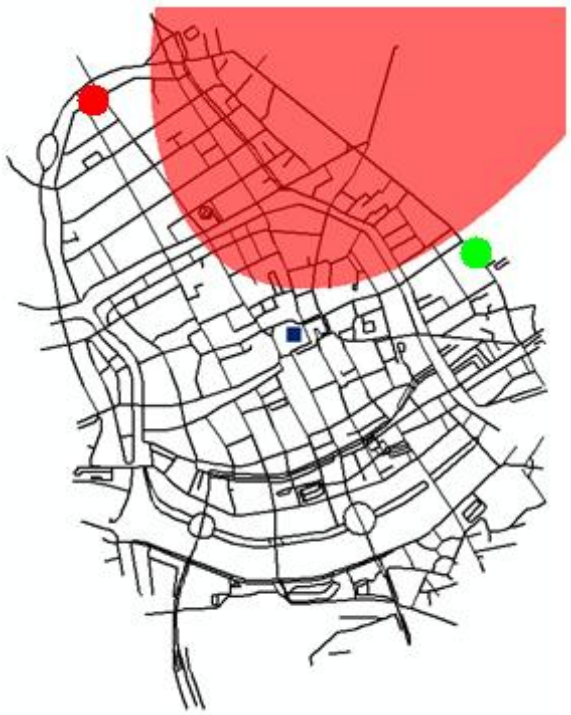


Figure 20: Case of Groningen

Reference route

The reference route for this case was, similar to the case of Rotterdam, created with the same algorithm as was used for the route determination in the other phases. As can be seen here the reference route (blue) starts by following a section of the ring road and changes to one of the access roads in a later stage. The routes calculated in the next two phases will be compared to the reference route.

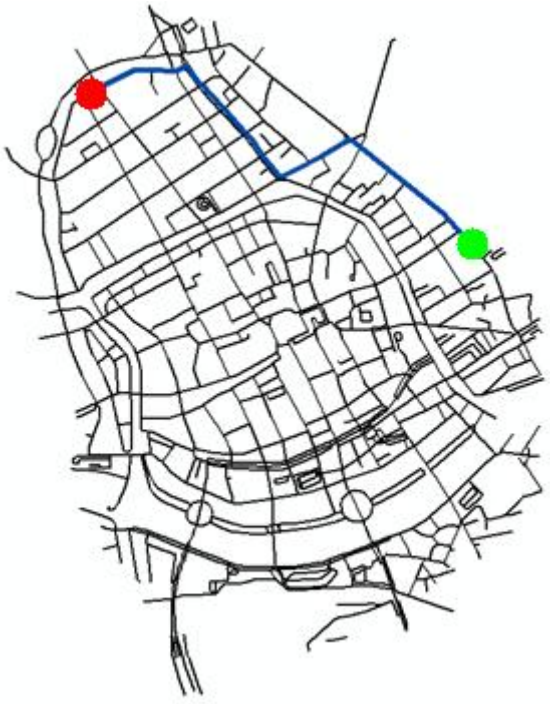


Figure 21: Reference route for the case in Groningen

5.3.3 Case 3: Delft

The case of Delft is a combination of the case of Rotterdam and the case of Groningen. In Delft the inner city is located on a manmade island, but the roads through the inner city and directly around it connect the north and the south of the subset in a relatively straight line. So the structure of the roads resembles the case of Rotterdam more, where the location of the inner city is more similar to the case of Groningen. In this case only closing bridges is less likely to have an effect on the eventual route since the destination (red) is also reachable without crossing the bridges around the inner city. These bridges are in reality unable to close for traffic. On the other hand there are a number of other types of temporal blocks in the inner city of Delft. The city has some moving cones that close off some of the roads in the city center. Some people are able to let these cones move down and pass with their vehicle. These cones can be opened at virtually every moment in time. It takes some time to have the cone move down completely. This adds to the total travel time. Though these automatic cones might at first sight seem to function as a temporal block, these only slow down the travel in practice. These cones are not considered here as temporal blocks, but as delays. The main reason is that these cones can be operated manually and the delay caused by these cones are independent of the time of arrival. This case will be extended by adding constant blocks, caused by road construction or a road block. This is done both to demonstrate the possibility to integrate constant blocks into the algorithm as well as force the route through the city centre. This will result in a balancing act between traveling extra distance or waiting for a bridge to open again.

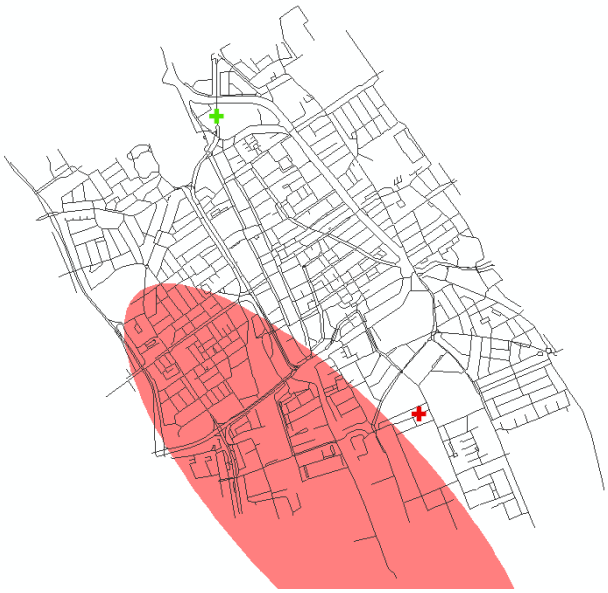


Figure 22: Case of Delft

Reference route

Also for the case of Delft a reference route was calculated using the algorithm that was used in all cases. The route leads the unit around the inner city using one of the connecting lines completely. This route is also covered by the plume. This helps to compare the results of the additions to the algorithm.

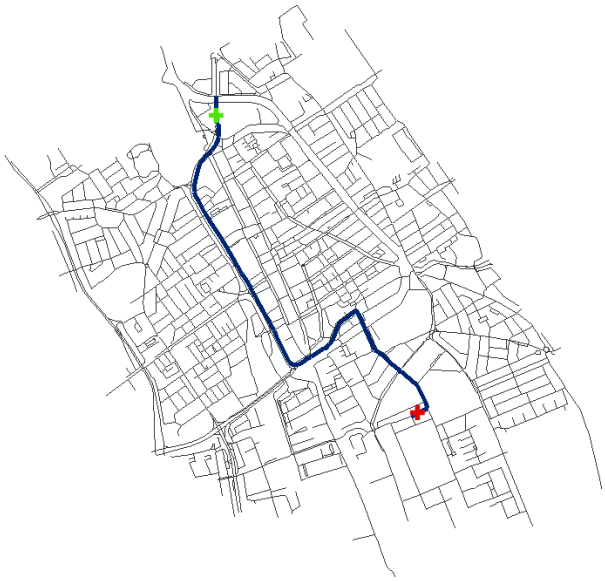


Figure 23: Reference route for the case in Delft

5.3.4 Case 4: Moving plume

In the cases above the dynamic data has all been related to the bridges. Using the same method a constant road block has been implemented in the third case. This case is intended to show that the implementation of a dynamic plume is similar to the implementation of the bridges and the implementation of the constant roadblock. The results will be compared with the reference route as is done in the other cases as well. The red plume applies to the time the route determination starts. The orange plume represents the same plume after a change in the wind direction and applies from a moment in the future. For comparison there will be two other routes that function as a reference. The first is the reference route that is created for all the other cases as well. The second is the route using only the first plume as a constant block

Figure 24: Case in Rotterdam with a moving plume

Reference route

Here the reference route is relatively straightforward. And not crossed by the outline of the plume. The route however will be altered since an additional threshold of 100 meters is used

for closing the roads around a plume. The other results will be compared to this reference route.

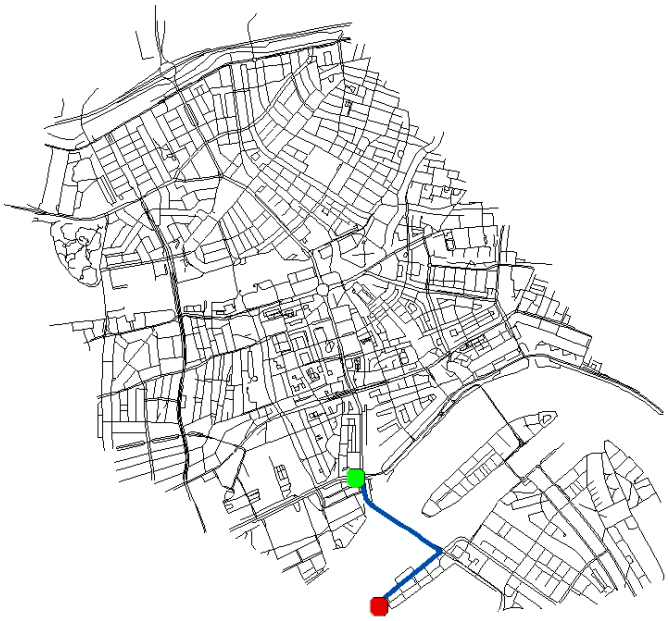


Figure 25: Reference route for the case in Rotterdam with a moving plume

5.3.5 Case 5: Destination inside the plume

In case of a disaster including toxics the exposure of people to these toxics should be prevented. It is however possible that an emergency unit will have to enter the restricted area to rescue some people who were left behind in the chaos for example. For cases like this, it should be possible to determine a route into the restricted area. Determining the least cost path in this case means that the plume, that cannot be avoided, should be traversed as short as possible. Therefore this case represents a situation where the destination (red) is inside the plume. The start point (green) is positioned in a way that leaves the position of entering the plume open. This provides the possibility to see what influence the penalty applied to the roads underneath the plume has on a route that has to get into the plume itself.

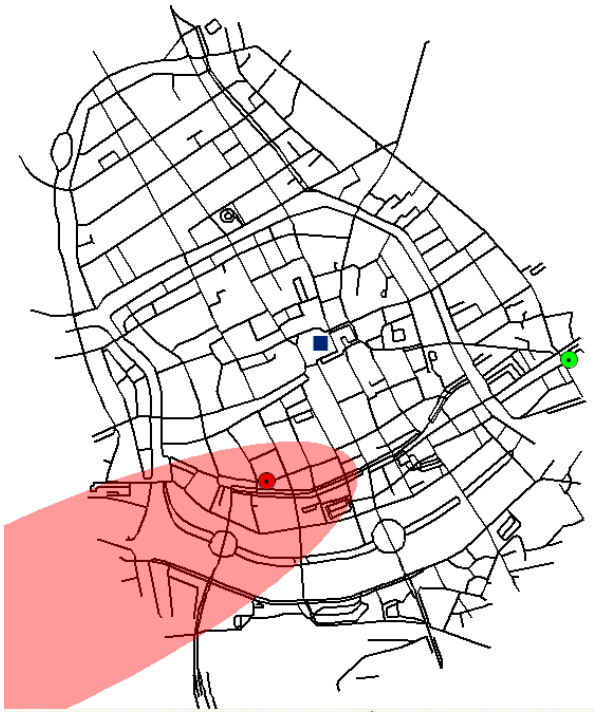


Figure 26: Case of Groningen with a destination inside the plume

Reference route

In this case the reference route is an almost straight line connecting the source with the destination point. It might be expected that the route changes when the plume is taken into consideration.

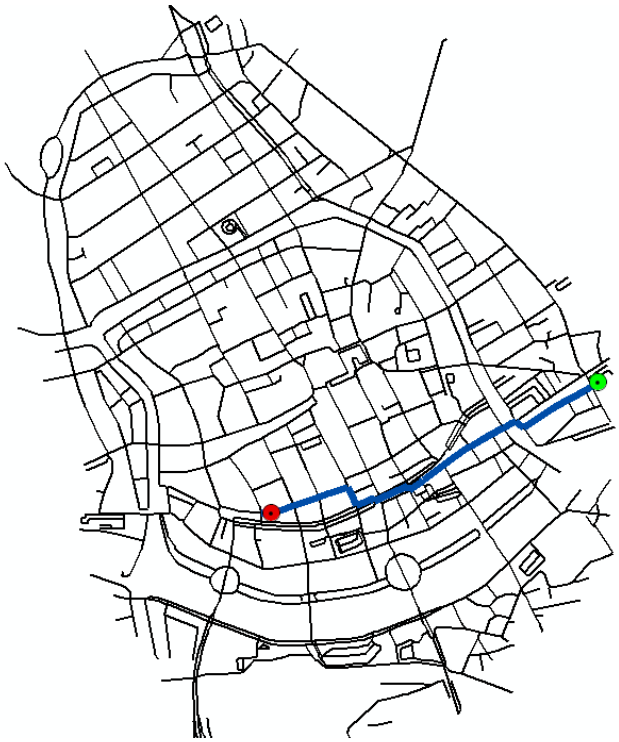


Figure 27: Reference for the Groningen case where the destination is inside the plume

6. Building an algorithm for static route determination

6.1 Introduction

Routes similar to the reference routes can be generated using normal route planners as these are in use today. The difference of this algorithm compared to the normal route planners is twofold. These two main differences are represented in the two separate phases defined in this research. The first difference is that it is possible to define a restricted area, that should not be crossed by the route unless the destination or the source is inside the area. This can be handled by the normal Dijkstra algorithm by altering the travel costs. The routes resulting from evading the restricted area will be described in this chapter. In the next chapter the second difference of the algorithm will be implemented and described. That chapter focuses on the incorporation of dynamic data into the algorithm. In the first section the algorithm for the incorporation of the restricted area into the route determination process will be described. In the next sections the results for the different cases will be examined. The comparison of all the results against the reference route will take place in Chapter 8. The textboxes presented in this chapter contain fragments of the actual code written in python.

6.2 Building the algorithm

For the calculation of the routes an algorithm was produced, based on the Dijkstra algorithm. To keep the algorithm relatively simple, the actions needed for the calculation of the route were split up into different functions. The algorithm consists of these functions, which are called in a particular order.
The algorithm starts with collecting some basic information and preparing the data and environment. This part of the script is depicted in textbox 3. Using the ArcGIS geoprecessor object (ArcObject) some environment settings are made. The first environment setting is set for overwriting the output. This allows to safe the route calculated in a shapefile. This shapefile will then be overwritten every time the algorithm is run. To achieve this the "overwriteoutput" is set to 1. In the next step the default folder for finding and saving the data is set. This is the workspace environment setting.
After the environment settings have been made, the input information is specified. Here the files that contain the source and destination points are specified for example. All these files receive an internal name, to call the information later in the script. Now all the information is hard-coded in the script, but these base information variables can also be used to give the user the possibility to specify its own input before each run of the script.
In the next steps the data itself is prepared using custom functions.

```
1.  import arcgisscripting, string

2.  gp = arcgisscripting.create(9.3)

3.  gp.overwriteoutput = "1"                # environment setting: do overwrite the
                                            # output
4.  gp.workspace = r"D:\Dijkstra"           # environment setting: default folder
5.  roads = "roads.shp"                     # variable with input data
6.  sourcepoint = "Startpoints.shp"         # variable with input data
7.  destpoint = "Endpoints.shp"             # variable with input data
8.  plume = "plume.shp"                     # variable with input data
9.  UnitID = 1010                           # variable with input data

10. # set the traveltimes of the roads in the network
11. AlterTraveltimes(roads, plume)

12. # determine the line ID's of the lines the source and destination location are
    located on

13. source = determineLine(UnitID, sourcepoint)
14. dest = determineLine(UnitID, destpoint)
```

Textbox 3: Basic settings and preparations

The first function (AlterTraveltimes) calculates the new travel costs per road segment. These travel costs are altered based on the location of the plume. In textbox 4 this function is shown. As can be seen the function consists of three ArcGIS tools. One can verify this because the toolname is preceded by “gp.”.

```
1.  def AlterTraveltimes(roads, plume):
2.      gp.MakeFeatureLayer(roads, "rlayer")
3.      gp.SelectLayerByLocation_management("rlayer", "INTERSECT",
            plume, "100 meter", "NEW_SELECTION")
4.      gp.CalculateField_management("rlayer", "MINUTES", "[St_Travel]* 4")
```

Textbox 4: custom function to alter the travel within the restricted area

First the roads are prepared for a selection in line 2, followed by a selection of these roads in line 3. This selection of roads is based on their intersection with the plume. All the roads that are underneath the plume or are within 100 meters of that plume are selected. The third tool, in line 4, multiplies the travel times of the selected road segments with a factor (e.g. a factor of 4). The determination of the factor was done in this phase by performing some small tests using the standard Network Analyst extension available in ArcGIS. The test results are described in textbox 6.

After the calculation of the new edge travel costs based on the location of the restricted area, the starting road segment and the destination road segment needs to be determined. The source and destination of the route is depicted by points by default. Therefore a function was

created that searches for the road segment that has a point on it. The function called in the algorithm is depicted in textbox 5.

```
1.  def determineLine(UnitID, point):
2.      gp.MakeFeatureLayer_management(point, "sourcepoint", "Id = " + str(UnitID))
3.      gp.MakeFeatureLayer_management(roads, "linesLyr")
4.      gp.SelectLayerByLocation_management("linesLyr", "INTERSECT",
            "sourcepoint")
5.      scur = gp.searchcursor("linesLyr")
6.      lineList = []
7.      row = scur.next()
8.      while row:
9.          lineList.append(int(row.getValue("ID")))
10.         row = scur.next()
11.     lineID = lineList[0]
12.     return lineID
```

Textbox 5: function to find the road segment that has a specified point on it.

In this function the points and the roads are prepared for selection in the first two lines of the function. In the third line the road segment, that is underneath the point, is selected. In the next lines of the function the possibility that the point is on an intersection is considered. Here all the road segments are stored in a list. In line 11 of the function only the first of the lines is selected and returned. So even though there could be more lines in the road dataset intersecting with the source or destination point, there is only one line that is returned. This line is used in the remainder of the algorithm as the first or last line of the route.

The output of the function is the ID of the road segment the start- or endpoint is on. This ID will be used to look-up information on travel costs, finding the connected roads and building a statement to export a route dataset.

Implementing the restricted area

To determine in what way the restricted area can best be evaded, different approaches were tested. To compare the approaches to closing the area, the destination of the route is located inside the plume. This is a situation that only exists in the fifth case considered in this research.

Consider the following case. A starting point in the south and a destination in the north. The destination is located inside a plume. The routes are calculated using a geometric network, created with ArcGIS Network Analyst.

Figure 28: Normal route determination without limitation on travel through the plume

The reference route (figure 28 right) obviously runs straight through the plume since the plume is directed towards the start point. On top of that the plume is not yet considered in the route determination process. The route created here is the fastest route possible between the start- and endpoint along this network. Question now is what the best method is to arrive safely at the destination. This means that the travel time within the plume should be minimized.

Blocking the plume

A first method would be to block all the roads entering the plume. This can be done easily using standard tools of the Network Analyst extension in ArcGIS. When all the roads into the plume are blocked, there is no route leading towards the destination, since the destination is inside the plume. One option would be to search for the blocks with the smallest Euclidian distance towards the destination point. When the closest eight blocks are deleted there is a fair chance that there will be a route possible towards the destination. The route determined in the case that eight blocks are deleted, does not deliver the desired results.

Textbox 6: Implementing the restricted area

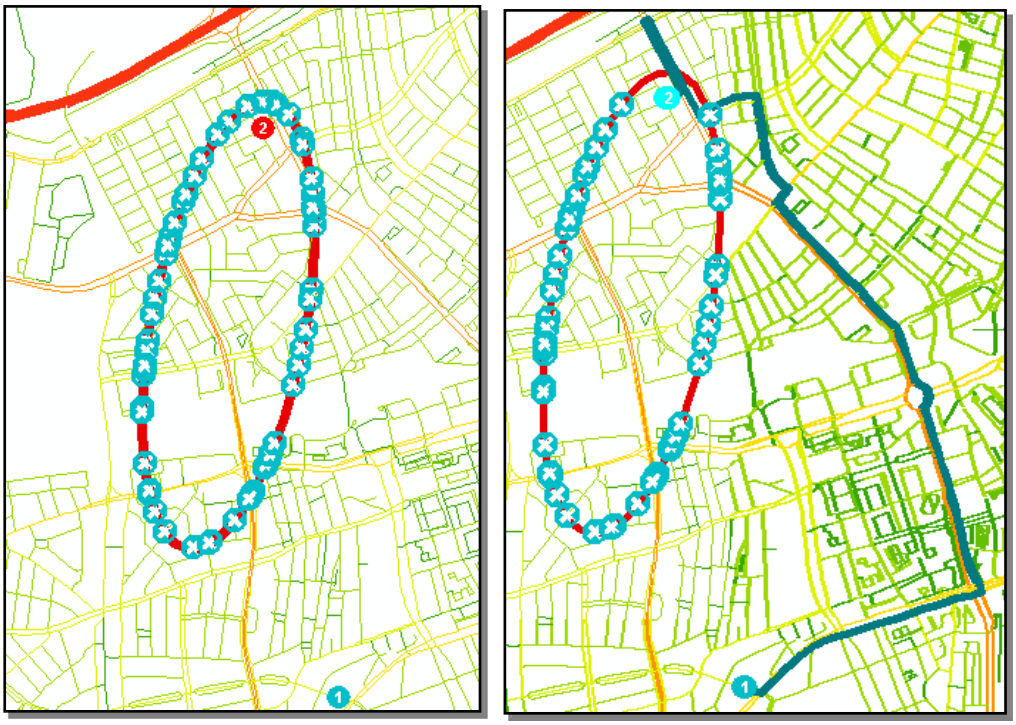


Figure 29: Route determination with blocks around the plume

The problem with this approach is that one cannot predict the number of entrance roads into the plume. If there are only eight entrances, deleting eight blocks would mean that the entire plume is opened. Basically this results in a route that is the same as the route determined without the plume. In another extreme case the eight closest blocks could all be positioned on a dead end street, not giving the opportunity to arrive at the destination. As a third problem the majority of the access roads could be concentrated on the far end of the plume. Meaning that though only a part of the blocks are deleted, the route could still cross the entire plume.
So in summary this approach does not deliver a solution for the problem to keep the route outside of the plume as much as possible. This is caused by the fact that by using the blocks, there is no limitation other than the block to find the shortest path through the plume. If the block happens to be deleted, because it is one of the closest eight blocks for that plume the route can run through the plume without encountering an additional penalty. Another approach would be to change the travel times of the road segments inside the plume.

Raising the travel costs within the plume
When raising the travel costs of all road segments within the plume, there is a penalty for traveling over every single road segment within the plume. Not only does it pay off to travel around the plume, but it also pays off to take the shortest route into the plume towards the destination. This means that depending on the magnitude of the penalty the chance is relatively small that the route will run through the plume if the destination is outside of the plume. The chance however always remains that at some point the route will be calculated through the plume, simply because the access roads into the plume are open. Question now remains what kind of a penalty should be introduced. This is not a factor that can be calculated. It depends on the ratio between the distance or costs of the shortest route and the distance or costs of the alternatives available. This ratio depends on the roads that are available, the type and concentration of the gas in the plume, the layout of the road network in the area, the distance between the source and the destination and the differences in costs per meter between the roads.

If there are for example two roads towards the destination. One road that allows one to drive 120 km/h and another road that only allows 20 km/h as a maximum speed. Adding a penalty to the fastest road will not automatically lead to another routing decision. The faster road might well deliver a faster route despite the additional penalty. If the factor is too high on the other hand it will take longer for a route to be found, when the destination is inside the plume. This is caused by the design of the Dijkstra algorithm itself.

Assume that the destination is within the plume and that the access road into the plume has a cost of 1 minute. In addition the travel time from the source towards the access road is 2 minutes. If the penalty for that road segment entering the plume will be set to 99 times the travel time, the new cost of that access road will be 99 minutes. So if you want to move into the plume it will cost you 99 minutes plus the 2 minutes it will take you to get to that access road. The total travel cost of the travel to the end of the access road will is 101 minutes. Because the Dijkstra algorithm has a greedy approach and explores the edges connected to the line with the lowest total travel costs, one can see that it will take a considerable amount of exploration steps before the access road into the plume would become the edge with the lowest total costs. The edges connected to the access road will only be explored as all the other loose ends of the explored network have a higher travel cost than 101 minutes. This simply is an performance issue, but since the route needs to be determined fast it is an important consideration for the design of the algorithm.

The above demonstrates that choosing the factor for the penalty is a balancing act. The approach here is therefore one of trying a few numbers and see what the results are. The factor however can be changed later as the results of the cases are examined. The picture on the left depicts the result when a factor of 2 times the travel costs is chosen. This result is obviously not the optimal result. Due to the one way traffic restriction on the data the route runs twice through the plume despite the penalty. In an optimal situation the route would have moved around the plume first and into the plume from the north towards the destination. The factor was therefore changed to 4. The result of using this factor is depicted in the picture on the right.

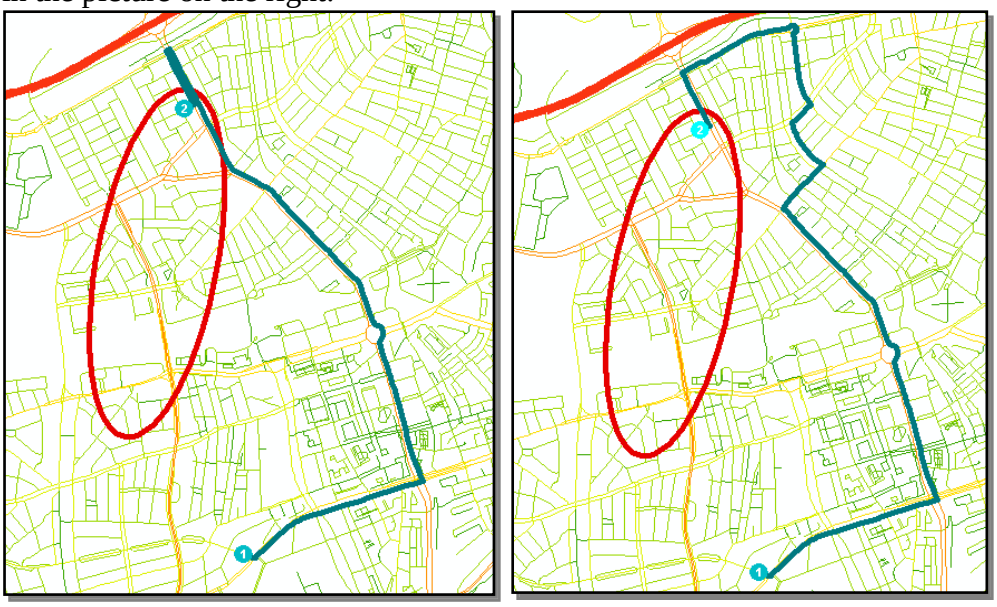


Figure 30: Altering the costs of travel with a factor 2 (left) and 4 (right)

The case where the factor of 4 is used delivers the expected result. The route moves around the plume and enters the plume in the north and moves downwind through the plume for a very short distance. Based on this result a factor of 4 is used for determining a penalty for now. Based on the results of the cases, as has been mentioned earlier, this factor might be altered.

Textbox 6: Implementing the restricted area (continued)

After establishing the source and destination edge of the route, the actual algorithm can start to determine the best route connecting the two. In order for the algorithm to work however some additional dictionaries and lists have to be created.

```
1. ## create the lists and dictionaries needed during the determination of the route
2. Graph = createRoadDictionary(roads)
3. queue = []                    # list
4. dests = {}                    # dictionary (key-value pair)

5. print "road segments collected"

6. ## set initial values; the line the source is on gets a travel cost of zero and has no
   ## predecessor
7. ID = source
8. dests[ID] = [0, 0.0]
```

Textbox 7: creating dictionaries and lists for the algorithm

The creation of these lists and dictionaries is depicted in textbox 7. A dictionary is a list with a key. In this case the key of the dictionary in line 8 is the ID of the road segment. By storing the key one can quickly find the attributes attached to a particular road ID, without knowing which entry in the dictionary contains the information asked for.
The first dictionary (Graph) created contains all the roads and is the result of a custom function. This custom function is depicted in textbox 8.

```
a. def createRoadDictionary(roads):
b.     scur = gp.searchcursor(roads)
c.     srow = scur.next()
d.     G = {}
e.     while srow:
f.         G[srow.getvalue("ID")] = [srow.getvalue("F_JNCTID"),
               srow.getvalue("T_JNCTID"), srow.getvalue("MINUTES")]
g.         srow = scur.next()
h.     return G
```

Textbox 8: storing the road segments in a dictionary.

In this custom function a cursor (line b) is created which is set on the first row of the roads table (line c). For each row in this table an entry is created in the dictionary with the ID of the road as the key (line e and f). Since the entries in a dictionary consist of a key-value pair some additional information can be stored as the value. Here the value is a list of three attributes (line f):

1. The “from” vertex of the edge
2. The “to” vertex of the edge
3. The travel costs of the road segment

When the row in the table is added to the dictionary, the cursor is set to the next row (line g) until all the rows have been processed.

Basically this function copies the important attributes from a table towards a dictionary. Though copying data that is available already seems pointless, retrieving information from a dictionary is quicker than retrieving the same information from a table. After creating the dictionary with the roads and returning this dictionary, a list is created for storing the queue (line 3). The queue will be filled with road segments that touch the explored part of the road network. The road segment in the queue with the lowest total costs will be added to the dictionary called "dests". This dictionary contains the edges and the minimal costs to arrive at that edge. Additionally the preceding edge is stored here as well. This enables one to trace back the shortest path, when the minimal travel costs to the destination have been determined. In line 7 and 8 in textbox 7 the source edge is added to the dictionary called "dests". Because this edge is the starting edge of the route there is no preceding edge and the travel cost is set to "0.0".

Now all the input variables, lists and dictionaries have been created. The next step is the actual algorithm. The algorithm is depicted in textbox 9.

```
1.  while ID != dest:
2.          con = findConnectedRoads(ID, Graph, dests, queue)
3.          for x in con:
4.                  queue.append(x)

5.          min = MinQueue(queue)
6.          dests[min[0]] = [min[1], min[2]]
7.          queue.remove(min)
8.          ID = min[0]
```

Textbox 9: The algorithm

The algorithm is a loop that runs as long as the ID of the road segment currently investigated, initially the source edge, is not the same as the destination road segment (line 1). In the second line a custom function is called that searches for the road segments connected to the current road segment. This custom function is depicted in textbox 10.

```
a.  def findConnectedRoads(ID, Graph, dests, queue):
b.      junctions = Graph[ID]
c.      connected = []
d.      for x in Graph:
e.          tmplist = Graph[x]
f.          queueContains = Contains(queue, x)
g.          if tmplist[0] == junctions[0] or tmplist[1] == junctions[1] or tmplist[0] ==
                junctions[1] or tmplist[1] == junctions[0]:
h.              if len(dests) == 0 and len(queue) == 0:
i.                  print "error in input, source is not entered in destination set"
j.              elif (x != ID) and (not x in dests) and (queueContains == 0):
k.                  listP = dests[ID] ## list of values of the predecessor
l.                  listC = Graph[x] ## list of values for the segment under consideration
m.                  travelTime = float(listP[0]) + float(listC[2])
n.                  connected.append([x, travelTime, ID])
o.      return connected
```

Textbox 10: fuction that searches for the connected road segments

In this function the current road segment is looked up in the dictionary containing all the roads (line b). From this dictionary the “from” and the “to” nodes are used to select the connecting road segments (line d – g). Both the “from” and the “to” nodes are used for this selection since the vehicle is allowed to move in both directions over any edge. For every road segment in the dictionary of roads the “from” and the “to” edge are compared with the “from” and the “to” edge of the current road segment. All the road segments that either share a “from” or a “to” node gain access to the “elif” loop, except for the current road segment itself, a road segment that has already been entered in the set of destinations or a road segment that is already in the queue (line j).
To determine whether the connected road is already in the queue another custom function is called. This custom function returns “true” if the connected road segment is in the queue (line 4 and 5) and “false” if it is not already there (line 6 and 7). The custom function is depicted in textbox 11.
For the road segments entered in the loop the total travel time is calculated (line k- m) . This is done by retrieving the minimum total travel cost of the current road segment from the “dests” dictionary (line k) and the travel costs of the connected road segment from the road dictionary (line l). The two travel times are added up, resulting in the minimum total travel costs for the connected road segment (line m). A list with the ID, the minimum total travel costs and the ID of its predecessor are added to a parent list with the connected road segments (line n). In textbox 9 line 3 and 4 one can see that the next step is to add each connected road segment to the queue.

```
1.  def Contains(queue, ID):
2.      a = false
3.      for x in queue:
4.          if x[0] == ID:
5.              a = true
6.          else:
7.              a = a
8.      return a
```

Textbox 11: function that determines whether the connected road is the same as the current

The next step is to retrieve the entry in the queue with the minimum total travel costs and add that road segment to the “dests” dictionary (textbox 9 line 5-7). This means that the minimum total travel costs for another road segment has been determined. The road segment just added to the “dests” dictionary becomes the current edge (textbox 9 line 8) and is used in the next iteration of this loop.

When the destination edge has been reached the loop ends and the minimal travel costs to the destination has been determined. The next step then is to trace the route back using the predecessor of the destination edge. The code used to do this is depicted in textbox 12.

```
1.  print "collecting results..."
2.  ETA = dests[dest][0]
3.  path = ReturnRoute(source, dest, dests)
4.  print path

5.  selstat = SelectRoute(path, roads)
6.  print selstat

7.  ResetTraveltimes(roads)
8.  gp.MakeFeatureLayer(roads, "Rlyr", selstat)          #selecting the edge features
9.  gp.CopyFeatures_management("Rlyr", "route.shp")      #copy the selected features

10. del gp
11. print "route determined"
12. print "The total travel time is " + str(ETA) + " minutes."
```

Textbox 12: returning the shortest path

First the total travel costs from source to destination is stored, including the penalties encountered for traveling through a plume, in the "ETA" variable (line 2). To trace back the route from the destination towards the source edge another custom function was created (line 3). This custom function simply follows the trace left by storing the predecessor for every road segment. This can be seen in textbox 13.

```
a.  def ReturnRoute(source, dest, dests):
b.      route = []
c.      v = dest
d.      route.append(v)
e.      while v != source:
f.          v = dests[v][1]
g.          route.append(v)
h.      return route
```

Textbox 13: function that lists all the edges that participate in the shortest path

The predecessor of the destination edge is entered along with the destination edge into a list (line c). In a loop the predecessor of the destination is set as the current edge and the current edge is entered into the list of participating edges (line e – g). The loop continues until the current edge is the source road segment (line e). The resulting list of participating edges is returned (line h).
With these lines the actual algorithm has ended its computation. To display the route in ArcGIS however two more functions were added to the script. These functions select the edges that are part of the route and export these to a separate file. The first custom function is depicted in textbox 14.

```
a.  def SelectRoute(path, roads):
b.      a = int(path[0])
c.      selstat = '"ID" = ' + str(a)
d.      path.remove(path[0])
e.      for x in path:
f.          a = int(x)
g.          selstat = selstat + ' OR "ID" = ' + str(a)
h.      return selstat
```

Textbox 14: Building a selection statement

In the line b and c a start of the statement is made. With the loop the statement is extended for every edge in the list with participating edges, until all the participating edges have been processed (line e – g). The selection statement is returned (line h) and is used to select all the edge features and add them to a new route set. This is done after the travel times of the road segments have been reset to their original value (textbox 12 line 5 – 8). The selection and the copy were made using ArcGIS tools as can be seen in textbox 12 (line 8 and 9). The custom function used to reset the travel times is depicted in textbox 15.

```
def ResetTraveltimes(roads):
    gp.CalculateField_management(roads, "MINUTES", "[St_Travel]")
```

Textbox 15: resetting the travel times

This function simply contains one ArcGIS tool that copies the values from the “St_Travel” field back to the travel costs field (MINUTES). This makes sure that the next route determination process will be performed on a clean dataset.
The final steps of the script consist of printing information on the travel costs to the screen and deleting the geoprocessor object (gp) from the memory (textbox 12 line 10 – 12). The complete script can be found in appendix I.

6.3 Testing the algorithm

Using the script described above the results for the different cases can be explored and assessed. This will be done case by case, comparing the results found when taking the plume into consideration against the reference route. In the pictures there is a buffer displayed around the plume with a red line. The penalty for moving inside the plume is also applied to this buffer. This buffer (or security area) aims to deal with the uncertainty that might exist about the actual location of the plume.

6.3.1 Case 1: Rotterdam

In this case a route is calculated between a source on an island in the south and a destination in the north. The plume is located between the source and destination blocking a route that would run in a straight line from the source to the destination. The situation is depicted in figure 31 including the reference route (dark blue) and the resulting route when considering the plume (light blue) in the route determination process. Apart from a difference in the routes geometry, obviously the length of the route is different as well. The difference in travel time is hard to compare however since the additional time spent results in a smaller health risk for

the people travelling along the route. The travel time of the different routes are depicted in Table 1.

Table 1: Change in travel time when considering the plume (Case 1)	
Route	Total travel time (Minutes)
Reference route	5,625
Route considering the plume (factor = 4)	7,274

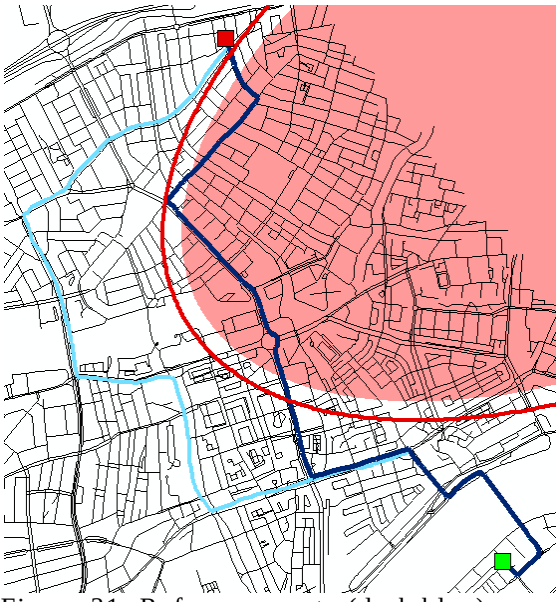


Figure 31: Reference route (dark blue) vs routing around the plume (light blue) (Case 1)

6.3.2 Case 2: Groningen

In this case both the source and the destination of the travel are on the edge of the subset. The reference route runs along one of the ring roads around the city centre. This ring road is however blocked by a plume. The route resulting from the algorithm that considers the plume is depicted in light blue in figure 32 along with the reference route (dark blue). Here it is clear as well that the plume has an important bearing on the result delivered. The difference in travel costs is depicted in table 2. The difference between the reference route and the route around the plume in this case is relatively large compared to the difference between the routes in the previous case. This is caused by the fact that that the start point is much closer to the plume here. This causes the route to move away from the plume first and move in the direction of the destination after the distance to the plume has increased.

Table 2: Change in travel time when considering the plume (Case 2)	
Route	Total travel time (Minutes)
Reference route	2,146
Route considering the plume (factor = 4)	5,165

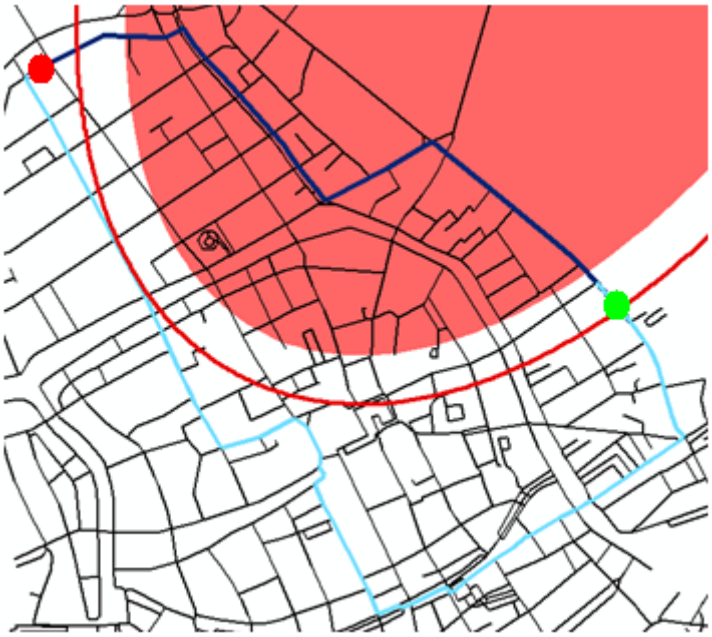


Figure 32: Reference route (dark blue) vs routing around the plume (light blue) (Case 2)

6.3.3 Case 3: Delft

In this case the source and destination are on the opposite sides of the city centre. Between the source and the destination there are a couple of connecting roads moving around the city centre on both sides and some of the connecting roads are moving through the city center. The plume in this case is positioned on one side of the city centre, blocking an important access road to the southern part of the city. The implications of this block are depicted in figure 33. Though based on the picture the difference between the reference route (dark blue) and the route around the plume (light blue) is considerable, table 3 shows that the difference in travel time is less than a minute.

Table 3: Change in travel time when considering the plume (Case 3)	
Route	Total travel time (Minutes)
Reference route	3,198
Route considering the plume (factor = 4)	4,054

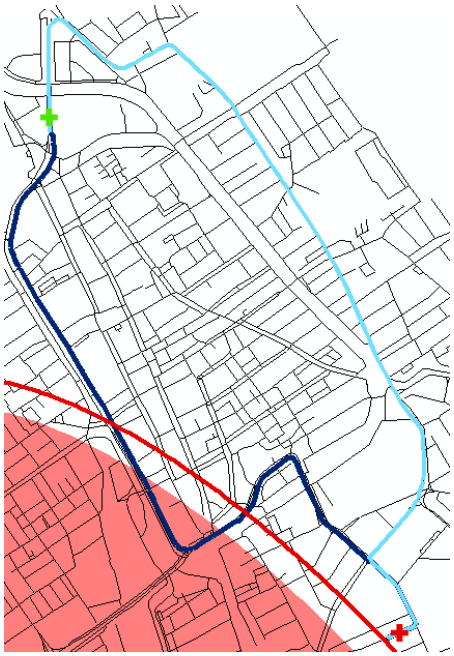


Figure 33: Reference route (dark blue) vs routing around the plume (light blue) (Case 3)

6.3.4 Case 4: Moving plume

The difference between this case and the other cases in this phase of the research is the length of the travel. The route to be calculated here is shorter than the other routes. In addition the number of alternatives to get from the start point to the destination is small. As can be seen in figure 34 this results in an undesired result. The reference route (dark blue) is determined as expected, but the route around the plume is not respecting the safety zone of 100 meters around the plume. The route calculated here is even passing through the plume, which imposes a health risk on the people within the vehicle being routed. The actual travel time of the newly calculated route is the same as the travel time for the reference route, as can be seen in table 4. This result indicates that the factor used to calculate the penalty for traveling through the plume or to close to the plume might well be too low. Because this result is considered unwanted the factor will be increased to a value of 10. As explained earlier raising this value has the drawback that the optimal route will take longer to be determined when the destination is inside the plume. It does however have the advantage that the chance that the route will move through the plume, as it did here, becomes smaller.

Table 4: Change in travel time when considering the plume (Case 4)	
Route	Total travel time (Minutes)
Reference route	2,348
Route considering the plume (factor = 4)	2,348 (costs: 4,646)

Figure 34: Reference route (dark blue) vs routing around the plume (light blue) (Case 4)

Because the results for this case indicate that a factor of 4 is insufficient to deliver the desired result the factor was increased to 10. The results using a factor of 4 will be compared with the results using a factor of 10 after the next case.

6.3.5 Case 5: Destination within the plume

The final case was created to see what the result would be if the destination was inside of the plume. The possibility to calculate the quickest route into or out of the plume is needed, because it could happen that the emergency response unit has to move into the area to evacuate some people that cannot get out of the area. On the other hand the emergency response unit might find that during measurements the plume moves over their heads. It is in such a case very convenient if the routing algorithm is capable of delivering the fastest route out of the area. As can be seen in figure 35 the route considering the plume (light blue) is very similar to the reference route (dark blue). Obviously this could mean that the reference route already followed one of the shortest paths into the plume. In this case however it needs to be considered that the destination point is relatively close to the northern edge. This means that the smallest possible time inside the plume can only be achieved by moving into the plume from the north-east.

Table 5: Change in travel time when considering the plume (Case 5)	
Route	Total travel time (Minutes)
Reference route	1,965
Route considering the plume (factor = 4)	2,131 (costs: 4,831)

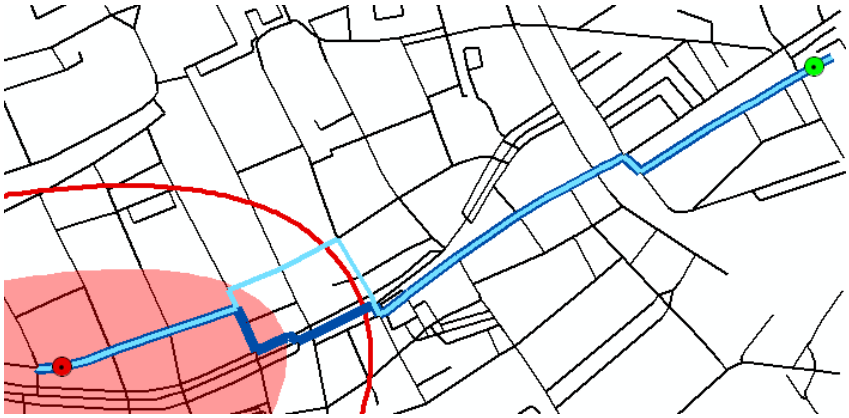


Figure 35: Reference route (dark blue) vs routing around the plume (light blue) (Case 5)

6.4 Adjusting the penalty factor

The factor used to increase the travel costs is the result of a balancing act between the additional time needed to travel around the plume and the increased risk associated with traveling through or close to the plume. During the cases presented above a factor of 4, which was established earlier, did not result in the desired result. Especially in the fourth case the distance to the plume became too small. This is caused by the fact that there is only one alternative available. The additional travel time associated with this alternative is relatively large, since the reference route is relatively short.
In the fifth case the route was almost the same as the reference route, where it was expected that the route would enter the plume in the north-east. Similar to the fourth case the alternative might result in a relatively high additional travel time compared to the total travel time of the reference route. This is caused by the layout of the roads giving access to the plume in the north. These roads are relatively long without any intersection with roads from the direction of the start point.
Though it is easy to just increase the factor used to determine the penalty on traveling near the plume, it should not have negative effects on the routes generated for the other cases. Therefore the script was adapted and tested on all cases. The first three cases are presented in the next section. In the two sections after that, case four and five are presented separately. In these three sections the new results will be compared to the results of the algorithm with a penalty factor of 4.

6.4.1 Implications of the new penalty factor on the routing results in case 1-3

In the first three cases the route created was already avoiding the plume. Any change in the algorithm should not affect the travel time of the routes for these cases. This means that the travel times found should remain the same. As is shown in table 6 this is the case here.

Table 6: Influence of increasing the penalty factor (Case 1 – 3)

Route\Total Travel Time (Minutes)	Case 1 Rotterdam	Case 2 Groningen	Case 3 Delft
Route penalty factor 4	7,274	5,165	4,054
Route penalty factor 10	7,274	5,165	4,054

In figure 36 – figure 38 the routes are geographically depicted in a yellow and a black line. As can be seen here the routes exactly overlap.

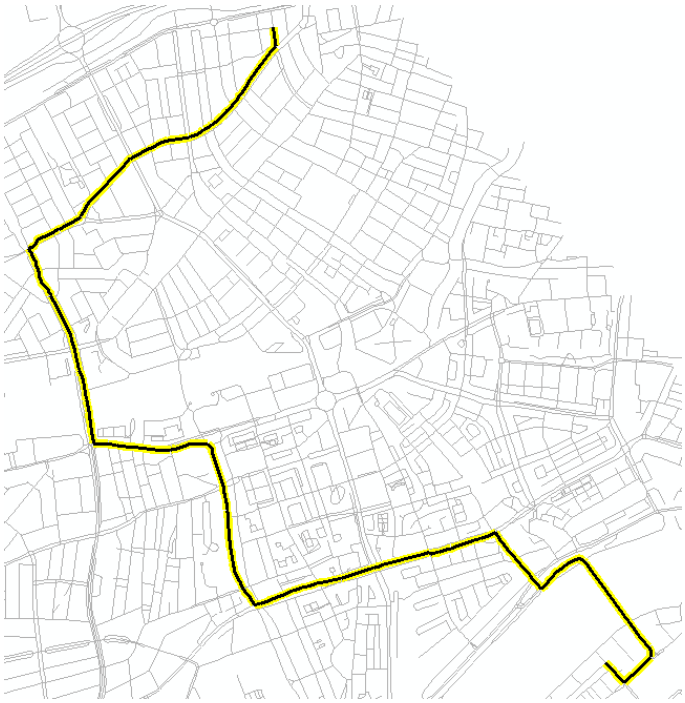


Figure 36: Penalty factor 4 (yellow) and 10 (black) in Rotterdam

Figure 37: Penalty factor 4 (yellow) and 10 (black) in Groningen

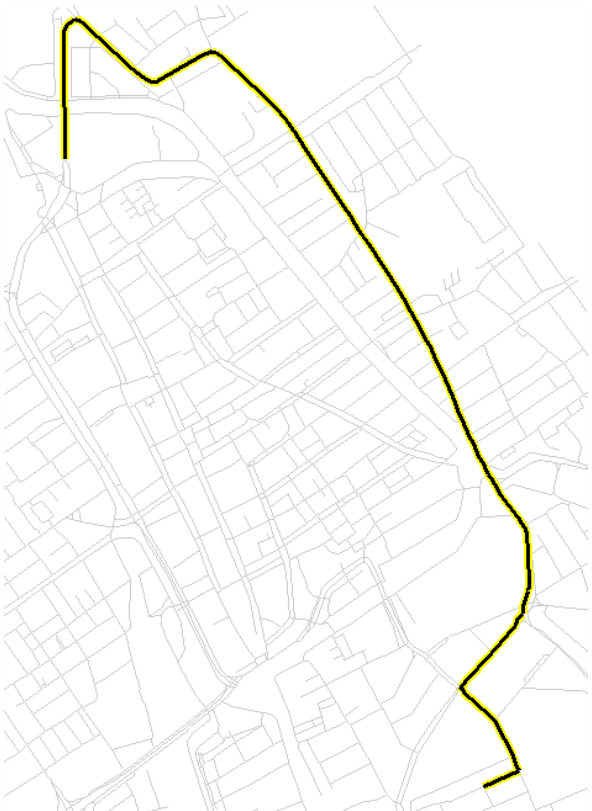


Figure 38: Penalty factor 4 (yellow) and 10 (black) in Delft

6.4.2 Implications of the new penalty factor on the routing results in case 4

The first penalty factor used produced a route that was considered undesired. The reason that the alternative was not found as the quickest route is that the alternative bridge over the Meuse is simply too far away. The penalty factor can be translated into the maximum accepted additional travel time. If a factor of 4 is used an alternative with an additional travel time of just under 4 minutes will be preferred over a route that is traveling through the plume for one minute. A similar line of reasoning can be set up for a penalty factor of 10. In this case a comparison between an additional travel time of just under 10 minutes and a travel time through the plume of 1 minute will be decided in favor of the additional travel time. As can be seen in figure 39, the alternative is desired over the penalty time in this case. In table 7 the travel times of both alternatives are shown.

Table 7: Influence of increasing the penalty factor (Case 4)	
Route	Travel time (Minutes)
Route penalty factor 4	2,378 (costs: 4,646)
Route penalty factor 10	4,958

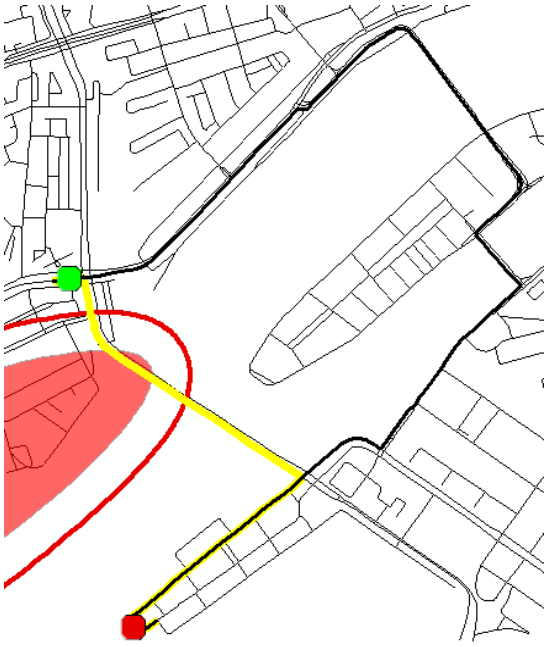


Figure 39: Penalty factor 4 (yellow) and 10 (black) around the Meuse

6.4.3 Implications of the new penalty factor on the routing results in case 5

In the fifth case it was expected or desired that the route considering the plume would enter that plume in the north-east. In figure 40 the change to the route as a result of the penalty factor is shown. Though the route does enter the plume in the north-east, it still moves west through the plume. This is caused by the security zone around the plume, which influence the travel west outside of the plume. In table 8 the difference in travel time is depicted.

Table 8: Influence of increasing the penalty factor (Case 5)	
Route	Travel time (Minutes)
Route penalty factor 4	2,131 (costs: 4,831)
Route penalty factor 10	3,470 (costs: 9,824)

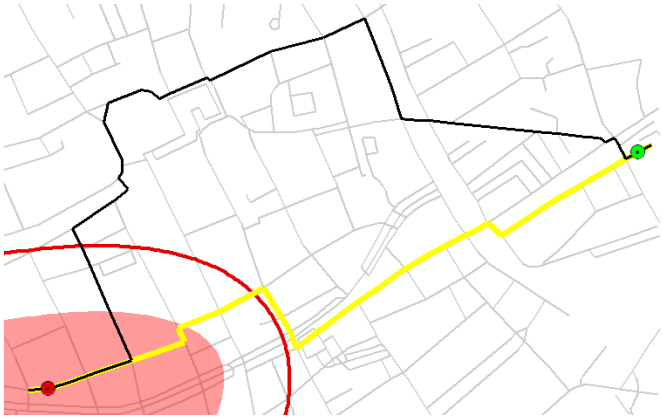


Figure 40: Penalty factor 4 (yellow) and 10 (black) with destination inside the plume

If it is assumed that the security zone is included in the actual plume, the additional security zone can be dropped. The implications of this assumption will be presented in the next section.

6.5 Removing the security zone around the plume

By removing the security zone around the plume the route is able to move close past the plume without receiving a penalty. In cases where the plume is between the source and destination, this is likely to result in shorter routes. In figure 41 – figure 45 the routes are represented. To be able to make a comparison the route calculated without a security zone (black) is presented together with the route calculated taking a security zone into consideration (yellow). Both routes were calculated using a penalty factor of 10.

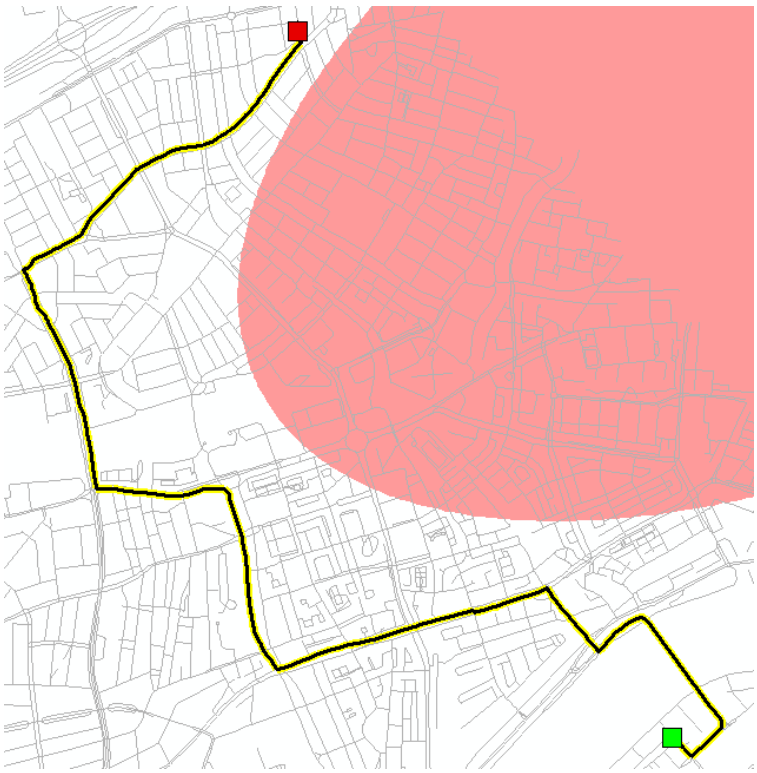


Figure 41: Case 1 with (yellow) and without (black) a security zone

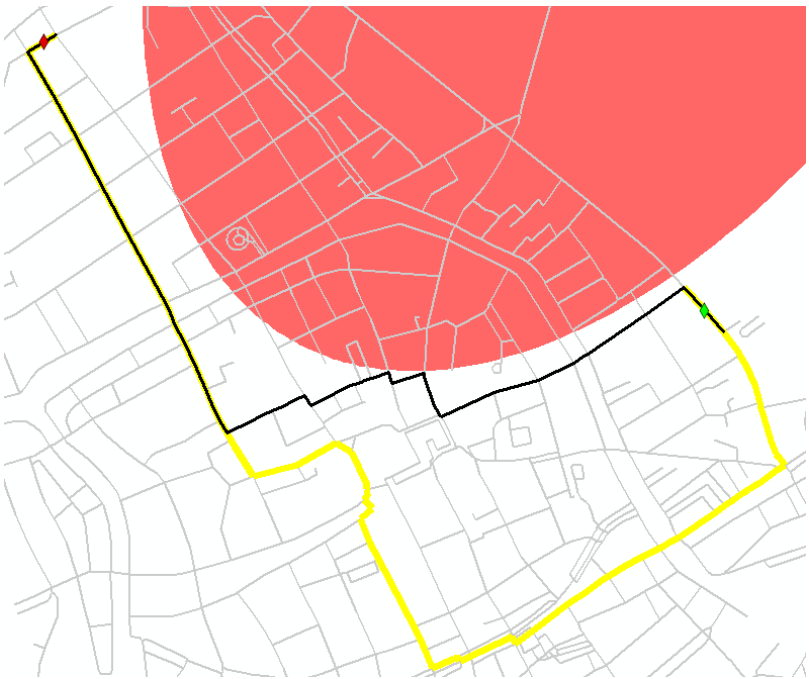


Figure 42: Case 2 with (yellow) and without (black) a security zone

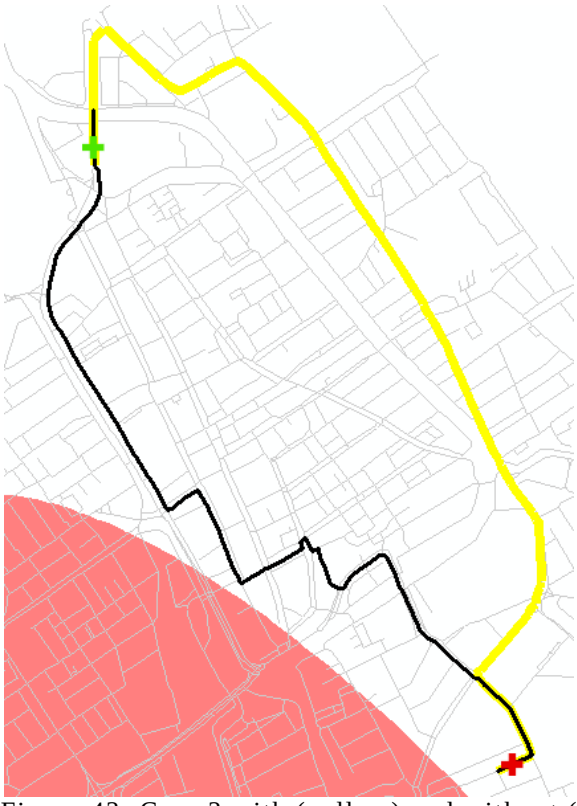


Figure 43: Case 3 with (yellow) and without (black) a security zone

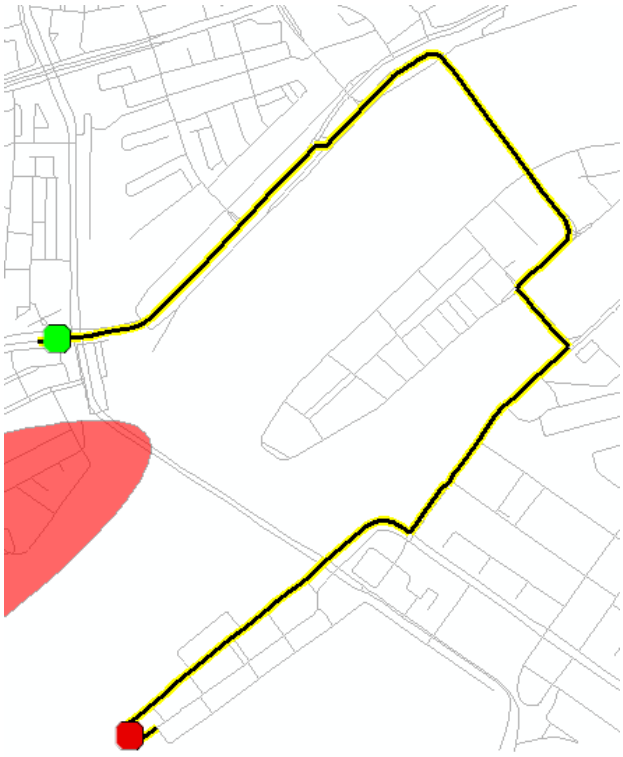


Figure 44: Case 4 with (yellow) and without (black) a security zone

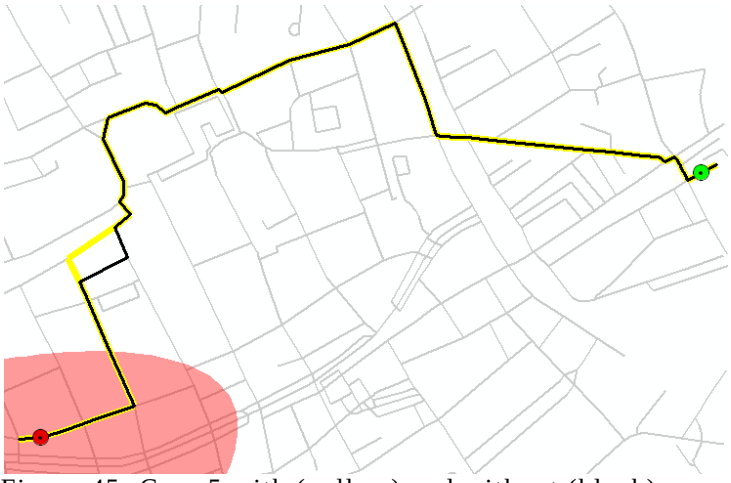


Figure 45: Case 5 with (yellow) and without (black) a security zone

From these pictures it can be deducted that the removal of the security zone had the largest impact in terms of route differences in the second and third case. The differences in travel time are depicted in table 9.

Table 9: Influence of removing the security zone (Case 1 – 5)					
Route\Travel time (Minutes)	Case 1: Rotterdam	Case 2: Groningen	Case 3: Delft	Case 4: Moving plume	Case 5: Destination within the plume
With security zone	7,274	5,165	4,054	4,958	3,470 (costs: 9,824)
Without security zone	7,274	3,718	3,796	4,958	3,389 (costs: 7,223)

As can be seen in this table the removal of this security zone has a small impact on the travel costs found. The routes generated here will be used for comparison against the routes that consider the dynamic data. The routes considering the dynamic blocks will be created and described in the next chapter.

7. Adapting the algorithm to incorporate plume and temporal block predictions and introducing the possibility to wait

7.1 Introduction

In the first phase the routes were calculated around the plume in a static manner. In this phase the algorithm will be extended to incorporate dynamic blocks. This means that all routes created here encounter a penalty for traveling through the plume and that the factor used to calculate the penalty is 10. First the extension of the algorithm will be discussed. In the section after that the results of incorporating the dynamic data are presented for each case.

7.2 Adapting the algorithm

For the algorithm to incorporate the dynamic data some changes need to be applied. The first thing that is needed is a custom function that can read the dynamic data from a file.
The custom function that was created is depicted in textbox 17.
Obviously such a custom function can only be created when the type of file and the storage of the data inside that file are known. In textbox 16 such a file is presented.

```
1,15280002378729,0.0,10.0
2,15280002378729,12.0,18.0
3,15280002378729,25.0,30.0
4,15280000797650,0.0,15.0
5,15280001446365,0.0,2.0
6,15280001446338,0.0,2.0
7,15280000682808,0.0,20.0
Autonumber, roadID,Closed from(minutes),Closed until(minutes)
```

Textbox 16: method of storage for the dynamic data

The bridges are stored in a comma separated value (CSV) file that contains one line for every event. This line contains the road segment ID of the bridge, the time the closing starts and the time the closing ends. The bridge ID is an integer with 14 characters, that need to be added manually to the file. The main advantage of this setup as opposed to storage in a shapefile is that it is easily read by the algorithm. It allows for the multiple storage of one bridge ID with different opening times.

```
1.  def waitTimeBridge(ID, time):
2.      waitingtime = 0
3.      t = float(time)
4.      ident = int(ID)
5.      bridges = open(r"D:\Dijkstra\bridges.txt")
6.      while 1:
7.          line = bridges.readline()
8.          brdg = line.split(",")
9.          if brdg[0] != "":
10.             if  int(brdg[1]) == ident and t > float(brdg[2]) and t < float(brdg[3]):
11.                 waitingtime = float(brdg[3]) - t
12.             else:
13.                 waitingtime = waitingtime
14.         if not line:
15.             break
16.     bridges.close()
17.     return waitingtime
```

Textbox 17: Reading the dynamic data from a file

In the first lines some values are set (line 2 – 4). The variable "waitingtime" represents the total waiting time associated with a segment based on the time of arrival. The time of arrival at the edge is stored as "t" and the road ID of the road considered is stored as "ident". After the file with the dynamic data is opened (line 5) the code checks every line in the csv-file to see whether the road under consideration is listed in the file (line 6 – 15). In cases it is listed and the time of arrival is within the timeframe that the road is closed the waiting time is calculated and returned. Now that the waiting time has been established it needs to be added to the total travel time of the road segment. This is done by adjusting an existing custom function. This custom function is depicted in textbox 18 including the adjustment made.

```
1.  def findConnectedRoads(ID, Graph, dests, queue):
2.      junctions = Graph[ID]
3.      connected = []
4.      for x in Graph:
5.          tmplist = Graph[x]
6.          queueContains = Contains(queue, x)
7.          if tmplist[0] == junctions[0] or tmplist[1] == junctions[1] or tmplist[0] ==
                junctions[1] or tmplist[1] == junctions[0]:
8.              if len(dests) == 0 and len(queue) == 0:
9.                  print "error in input, source is not entered in destination set"
10.             elif (x != ID) and (not x in dests) and (queueContains == 0):
11.                 listP = dests[ID]
12.                 listC = dict[x]
13.                 wait = waitTimeBridge(int(x), listP[0])
14.                 travelTime = float(listP[0]) + float(listC[2]) + float(wait)
15.                 # wait = waiting time for bridges
16.                 connected.append([x, travelTime, ID])
17.     return connected
```

Textbox 18: custom function adapted to incorporate waiting time

The place where lines were added are bold in textbox 18. The waiting time for the edge under consideration is entered into the custom function that reads the csv-file in line 13. The waiting time that is returned by the custom function is stored in the variable “wait”. This variable is used in line 14 to compute the total travel time to the end of the edge currently under consideration.
The rest of the algorithm was left unchanged. For all cases the adjusted algorithm was used and the results are presented in the next section.

7.3 Testing the adapted algorithm

7.3.1 Case 1: Rotterdam

Using the algorithm described above a number of bridges were closed for traffic. At first the bridges depicted in figure 46 were closed for traffic. The labels in the picture represent the time the bridge is closed for traffic in minutes. The travel starts at time 0.

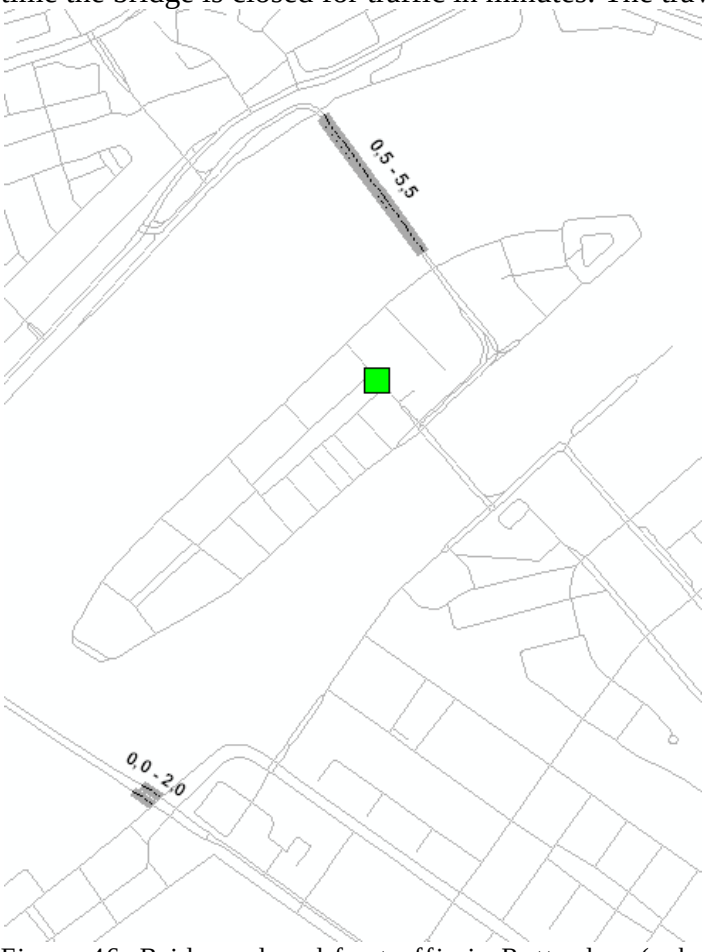


Figure 46: Bridges closed for traffic in Rotterdam (sub-case 1)

The resulting route taking the plume and the bridges into consideration is depicted in figure 47.

Figure 47: Resulting route (sub-case 1)

In the second sub-case some additional bridges were closed for traffic as is depicted in figure 48. The closing times of the bridge in the north have changed in this case. This bridge is closed twice for a shorter period instead off closing the bridge once for a longer period. The route calculated for this sub-case is presented in figure 49.

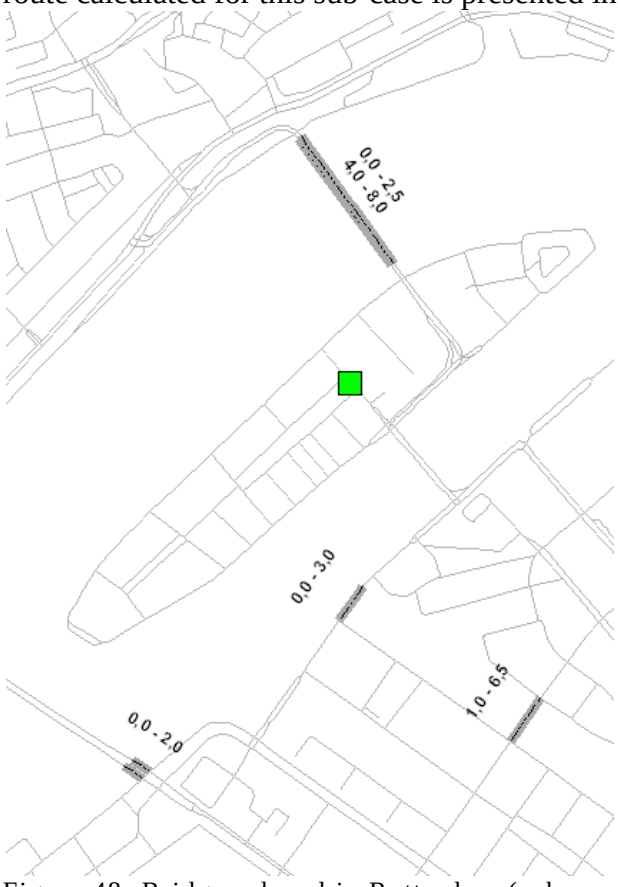


Figure 48: Bridges closed in Rotterdam (sub-case 2)

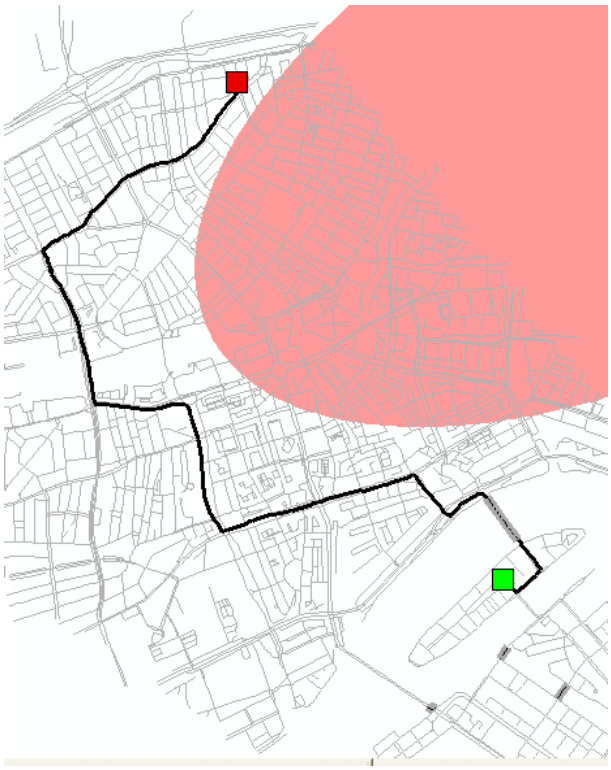


Figure 49: Resulting route (sub-case 2)

In both sub-cases the resulting route crosses a bridge that was closed for a particular amount of time. The difference in the closing times of the northern bridge and the addition of some bridge closings in the south result in a route that moves north instead off the south, as was the case in sub-case 1.
In table 10 the travel information for the routes created here is depicted. In this case there are no penalties involved for traveling through the plume.

Table 10: Driving and waiting time for the sub-cases of case 1

	Driving time	Waiting time	Total travel time
Sub-case 1	7,770	0,583	8,353
Sub-case 2	7,291	1,807	9,098

The resulting route of the second sub-case will be used in the comparison between the different routing algorithms.

7.3.2 Case 2: Groningen

For this case the algorithm was forced to balance waiting time with the additional travel time needed for each alternative. All the bridges that could be reached without crossing the plume were closed in the second sub-case. In the first sub-case however the most remote bridge with respect to the plume was left open. The bridges and their closing times for the first sub-case are depicted in figure 50.

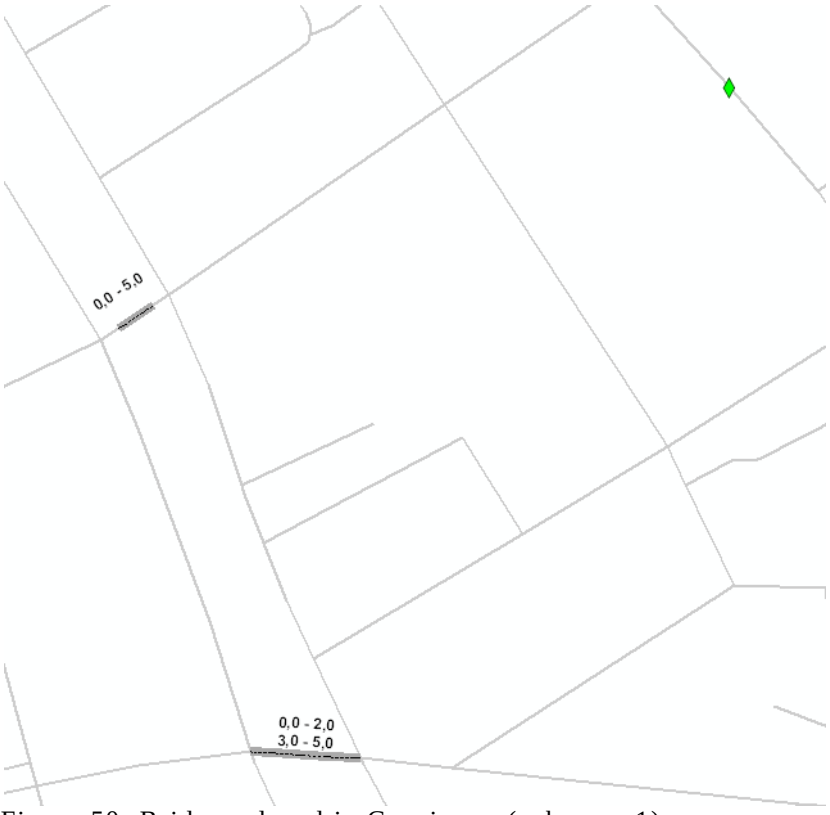


Figure 50: Bridges closed in Groningen (sub-case 1)

As can be seen here the lower bridge is closed for two periods of 2 minutes. This leaves the option open to move across that bridge between the closings. The bridge in the north is closed for a longer period. The result of the algorithm is depicted in figure 51. It can be seen here that the algorithm found a faster route by moving around the bridges and not use the option to cross the lower bridge in figure 50 between the closing.

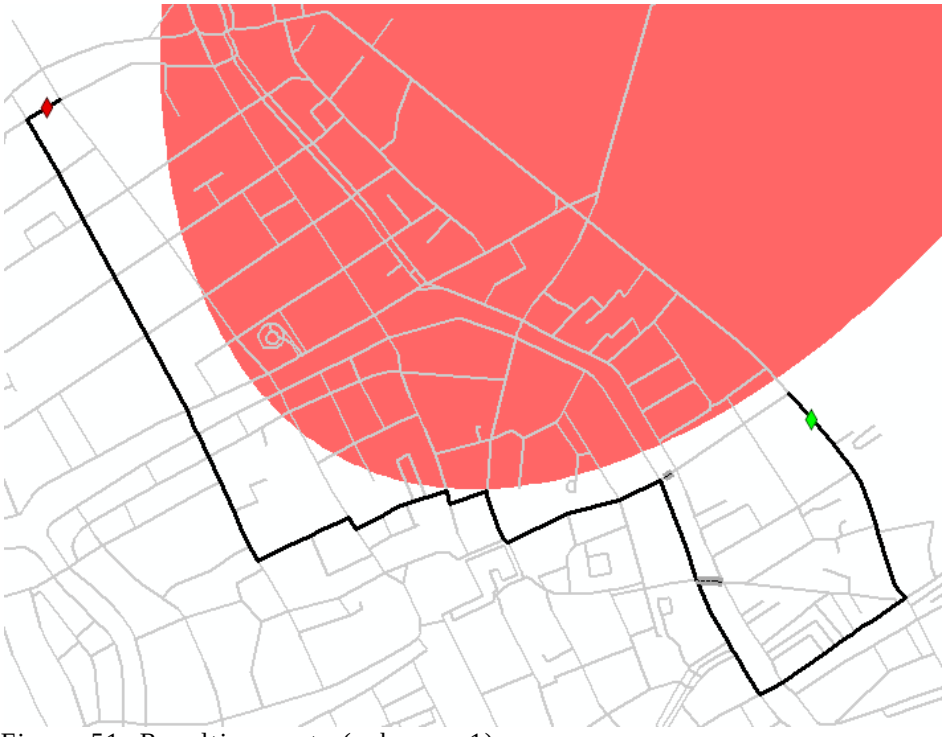


Figure 51: Resulting route (sub-case 1)

Since the route of sub-case 1 moves across the open bridge this bridge was closed in the second sub-case. The bridges and their associated time schedule for closing are depicted in figure 52.

Figure 52: Bridges closed in Groningen (sub-case 2)

The result of the balancing act the second sub-case calls for is represented in figure 53. Here it becomes clear that the balancing act results in waiting for the middle bridge and cross that bridge when it is open for a short period. The waiting and driving times associated with the two routes are shown in table 11. From the table as well as from the pictures above it becomes clear that the route in the first sub-case does not involve any waiting time as opposed to the route of the second sub-case. In the first sub-case the travel time accepted is higher than the travel time of the route of the second sub-case. The higher travel time however resulted in an avoidance of any waiting time, making the eventual route found shorter than the route in the second sub-case.

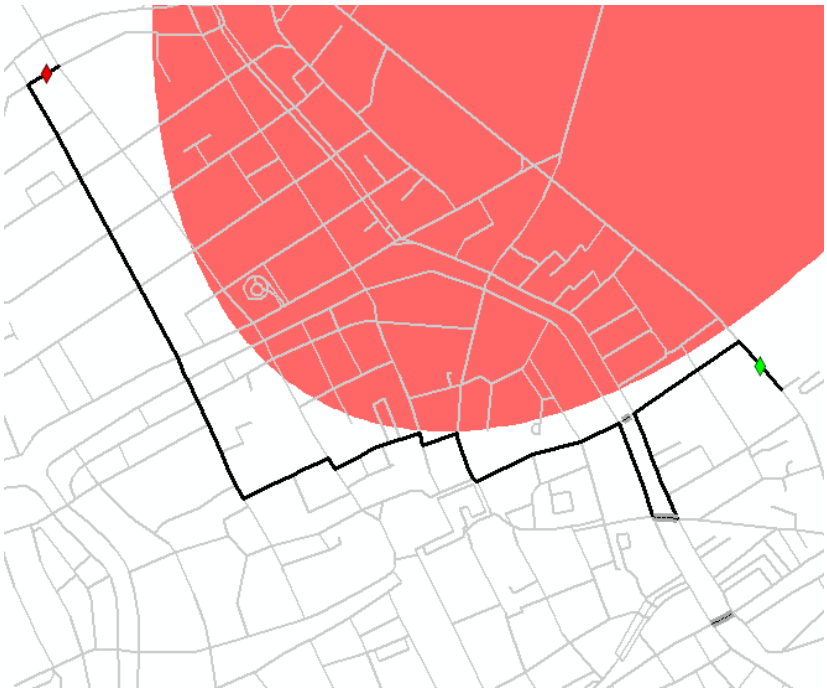


Figure 53: Resulting route (sub-case 2)

Table 11: Driving and waiting time for the sub-cases of case 2			
	Driving time	Waiting time	Total travel time
Sub-case 1	4,503	0,000	4,503
Sub-case 2	4,205	1,389	5,594

For the comparison made in chapter 8, the second sub-case will be used for this case.

7.3.3 Case 3: Delft

Delft is a city with relatively straight connecting roads between the northern and the southern part of the city. Due to the large amount of water in the city there are a number of key bridges connecting the southern part to the rest of the city. In the first sub-case one of these bridges is closed. In figure 54 this bridge is depicted including the period it is closed for. With this bridge being closed the fastest route from the source to the destination is depicted in figure 55.

Figure 54: Bridges closed in Delft (sub-case 1)

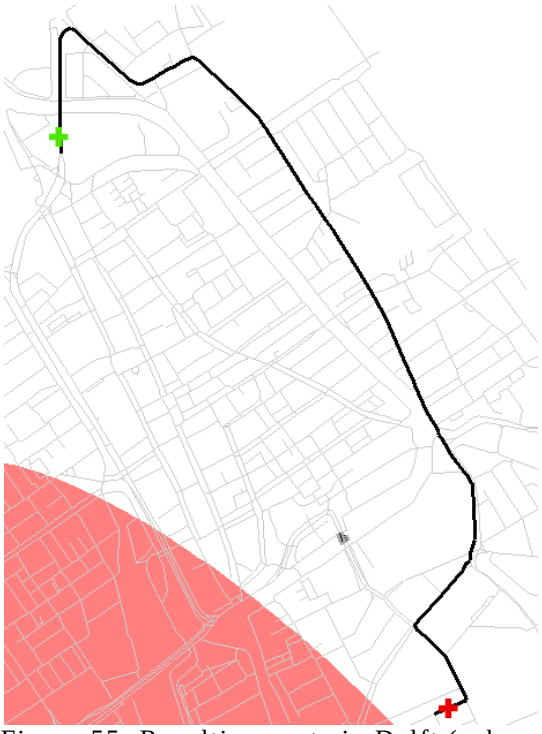


Figure 55: Resulting route in Delft (sub-case 1)

As can be seen here the route moves around the city center. In the second sub-case this route is closed permanently using a constant block. The constant block is created by blocking an edge from moment 0 until moment 99. This means that the edge is closed during the first 99

minutes of the travel. The constant block (red) and the new bridges added are depicted in figure 56.

Figure 56: Bridges closed in Delft (sub-case 2)

The resulting route in this sub-case is presented in figure 57. Though the routes seem to move around the blocks to avoid waiting time, but as can be seen in table 12 the vehicle being routed needs to wait for a small amount of time. Since the second sub-case is more complete in the amount of implemented bridges, this sub-case will be used for the comparison in the next chapter.

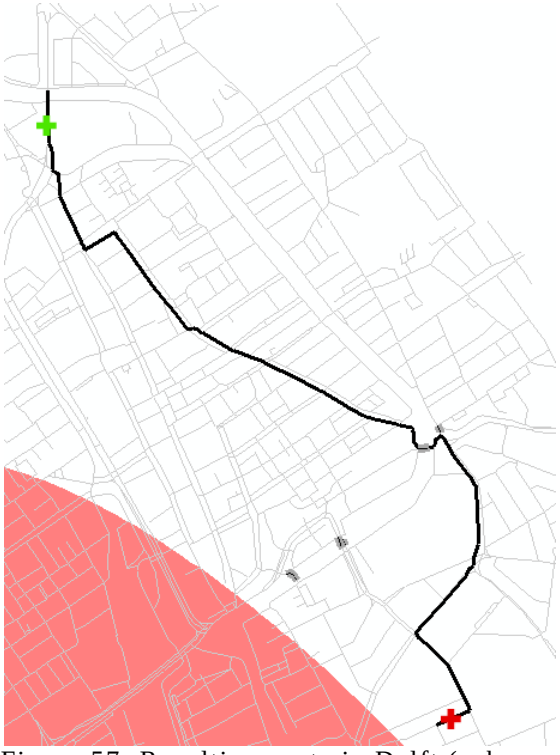


Figure 57: Resulting route in Delft (sub-case 2)

Table 12: Driving and waiting time for the sub-cases of case 3			
	Driving time	Waiting time	Total travel time
Sub-case 1	4,054	0,000	4,054
Sub-case 2	4,419	0,129	4,548

7.3.4 Case 4: Moving plume

This case is different from the other cases due to the fact that not the bridges will be part of the dynamic data, but only the plumes. The area the first plume covers will be closed for two and a half minutes. The area covered by the second plume will be closed from two and a half minutes after the start of the routing process until 99 minutes after the start of the same routing process. This value is the same as the value used in the previous case to close a road permanently. The closings are displayed in figure 58. To make sure that the route calculated produces the right result, a minor change had to be made to the algorithm. In all cases up to this point a penalty was applied to the area underneath the plume by multiplying the travel time with a factor. To prevent the plume to be taken into consideration twice, this factor was set to 1 instead of 10. Now travelling through the plume at the expense of the additional travel costs is no longer possible. A route using the roads that are covered by the plume will only be returned when it pays off to wait for the plume to wear off.

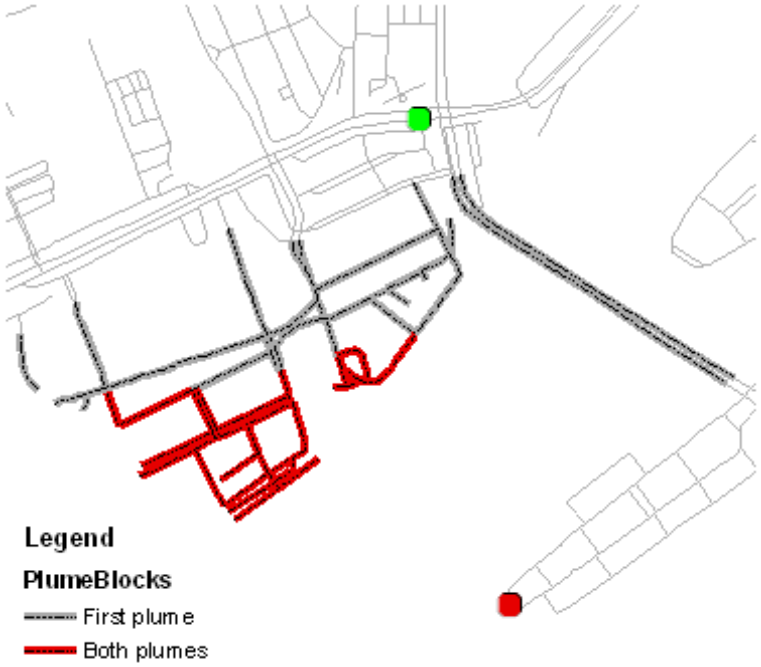


Figure 58: Road closings caused by the dynamic plume

The area affected by the plume locations is relatively large compared to the amount of closings presented in the earlier cases. The number of closings that have an effect on the route are on the other hand relatively small. The route that is calculated based on this dynamic data is depicted in figure 59. This route will also be used in the next chapter to compare against the results of the previous chapter.

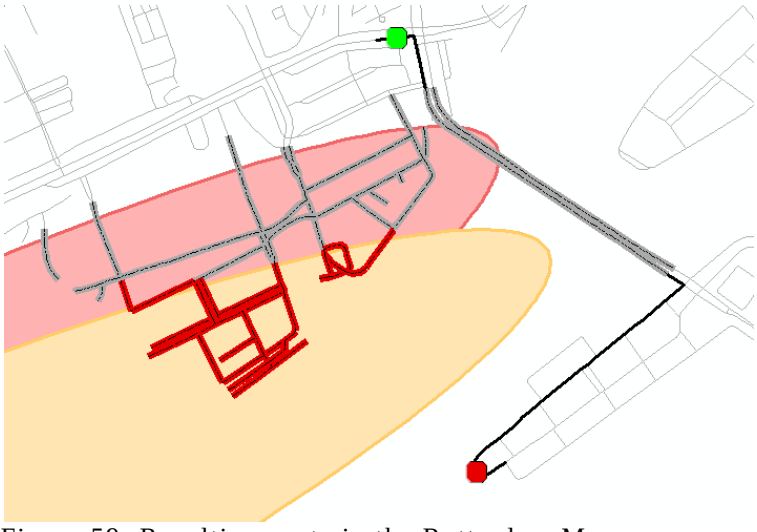


Figure 59: Resulting route in the Rotterdam Meuse area

The travel times associated with this route are depicted in table 13. Since this is the only route created for this case in this phase, it will be used in the comparison performed in the next chapter.

Table 13: Driving and waiting time for case 4

	Driving time	Waiting time	Total travel time
Moving plume	2,348	2,385	4,733

7.3.5 Case 5: destination inside the plume

In this case some of the bridges around the city center were closed to force the algorithm to choose between moving around the city or waiting in front of a bridge. In the first sub-case only one bridge was closed. This results in the obvious evasive route to get around that bridge. This closed bridge is depicted in figure 60 and the resulting route in figure 61.

Figure 60: Bridge closed in Groningen (sub-case 1)

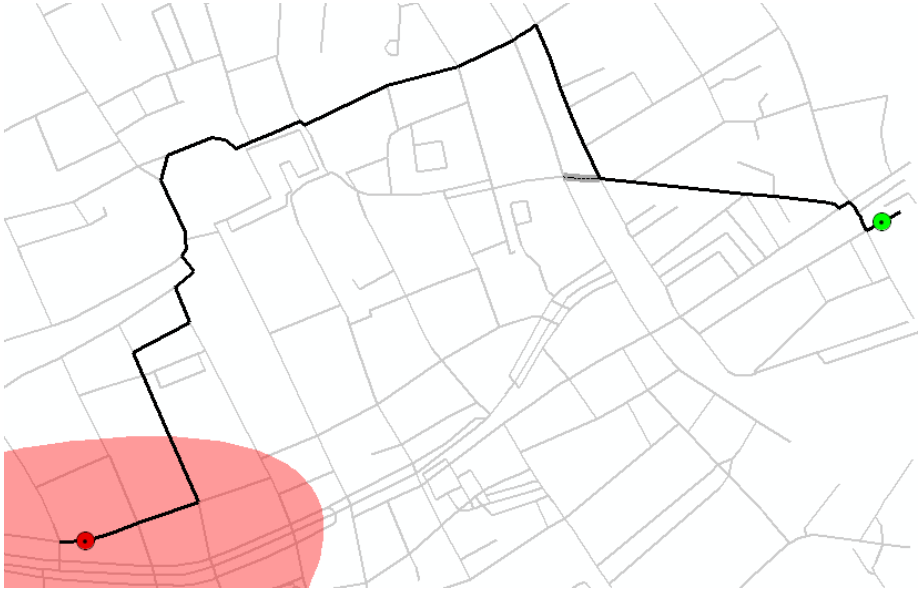


Figure 61: Resulting route with the destination inside the plume (sub-case 1)

In the second sub-case more bridges are closed as is depicted in figure 62, which close all the options for alternatives roughly located along a straight line between the source and the destination. As can be seen however in figure 63, it does pay off to take an alternative route over waiting in front of one of the bridges. The associated travel times are depicted in table 14. The result of the second sub-case will be used in the next chapter to compare the differences between calculating a route considering only the plume and considering the plume and the dynamic data.

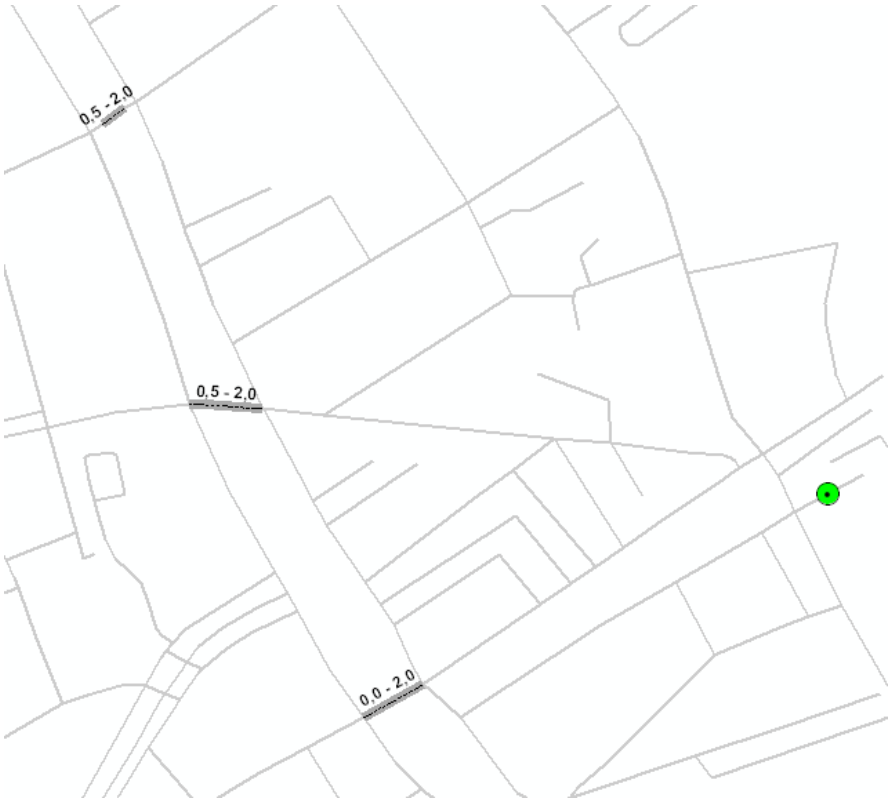


Figure 62: Bridges closed in Groningen (sub-case 2)

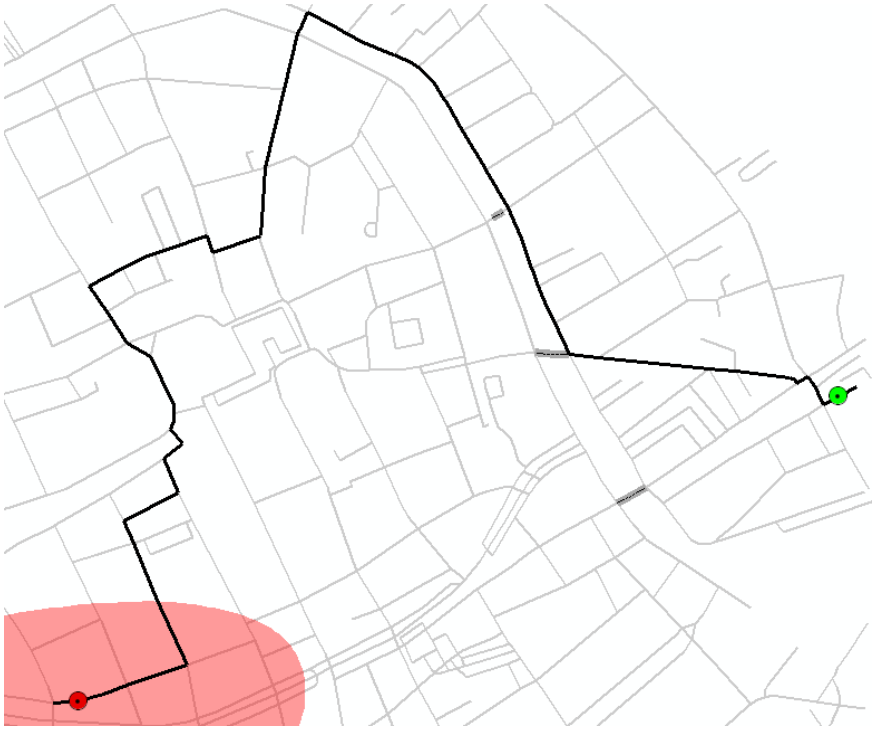


Figure 63: Resulting route with the destination inside the plume (sub-case 2)

Table 14: Driving and waiting time for the sub-cases of case 5			
	Driving time	Waiting time	Total travel time
Sub-case 1	3,404 (Including penalty: 7,238)	0,000	3,404 (Including penalty: 7,238)
Sub-case 2	3,722 (Including penalty: 7,556)	0,000	3,722 (Including penalty: 7,556)

8. Simple route determination vs route determination incorporating predictions

8.1 Introduction

Though the routes created for all the cases in the previous chapters represent the least cost route, the circumstances they were determined under are different. This resulted in different routes in terms of geography and also in terms of travel time and travel costs. In this chapter it is assumed that at one particular moment in time the two different algorithms were used to determine a route between the same source and destination. The difference in routes calculated in this case is not caused by different circumstances, but by incomplete input data. In this case it will help to investigate the advantages or disadvantages of using dynamic data in the algorithm. In the next section the differences between the routes will be compared per case. In all cases the route determined considering the bridge closing times are represented with a black line. The route not considering the bridges is represented by a yellow line.

8.2 Comparison of the Results

8.2.1 Case 1: Rotterdam

In the first case a route is determined through Rotterdam. As can be seen in figure 64, the routes determined are very similar. Both in the case the bridges are considered and in the case the bridges are not considered the algorithm chooses this route. As can be seen in figure 65 a small difference in route and as a result of that in the travel time is caused by a small difference in the route at the start.

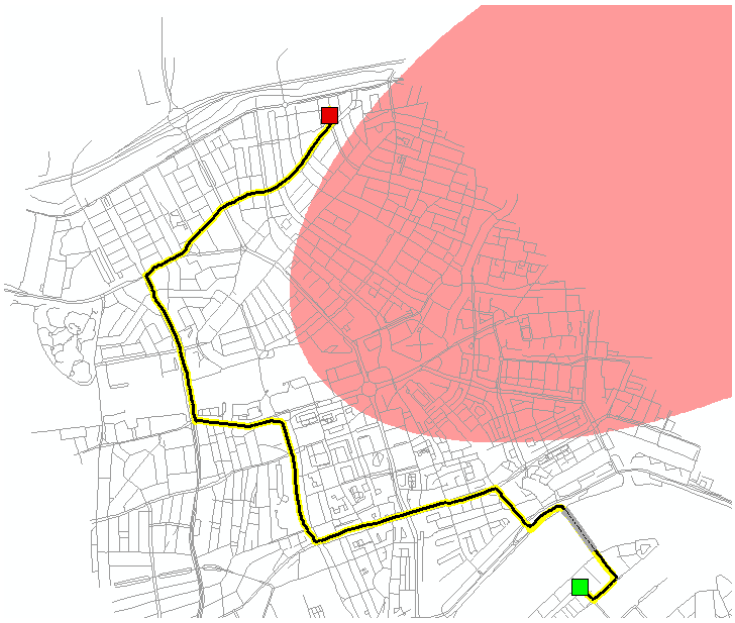


Figure 64: Route considering the bridge (black) versus normal route (yellow)

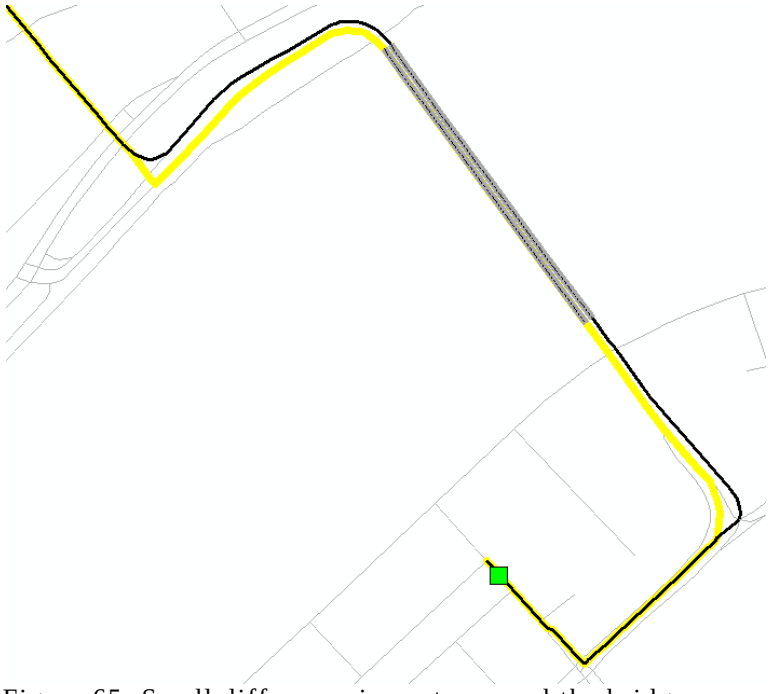


Figure 65: Small difference in route around the bridge

To represent the difference in travel times between the two routes a schematic overview is given in figure 66. In the left part of this schema the route considering dynamic data is represented in green. The horizontal axis represents the driving distance of the two routes. The green line represents the path of a vehicle and the vertical axis represents the travel time. When the green line runs vertically, this means that one has to wait here. The black lines casting a gray shadow over the schema are the bridges. As can be seen the green line runs vertically along the bridge, meaning that the vehicle only travels in time at that point. This means the vehicle waits.
On the other side the route is represented, that was calculated without considering the bridges. The small blue line represents the route that was returned by the algorithm and the red line is the travel of a vehicle that chooses to follow that route. In this case the arrival of the red and the blue line at the destination differs because the algorithm used to calculated this route did not consider the bridges.

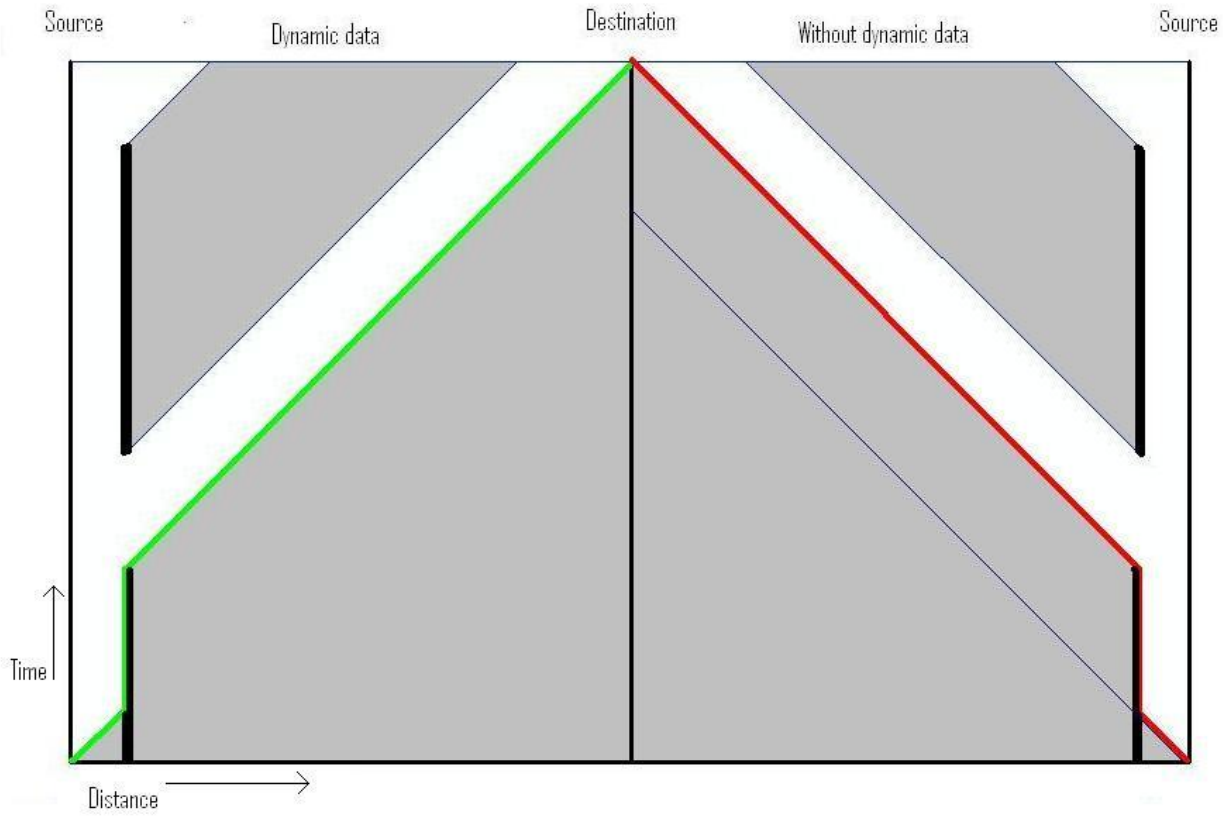


Figure 66: Schematic representation of the two routes

Table 15: Difference in anticipated travel time and actual travel time (Case 1)			
	Anticipated travel time	Waiting time	Actual travel time
Route considering predictions (black)	9,098	1,807 (expected)	9,098
Route without considering predictions (Yellow)	7,274	1,842 (unexpected)	9,116

The table above shows that even though the difference in travel time in this case is relatively small, the main difference here is that in the case the bridges are not considered the actual travel time is not known beforehand. In a sense this route is delayed due to the bridges whereas the route considering the bridges arrives at about the same time but was not delayed with respect to the predicted travel time. So in a case where no alternative route is generated the algorithm considering the bridges produces a better estimate of the trip length in minutes.

8.2.2 Case 2: Groningen

In the case in Groningen all relevant bridges were closed. This means that any route would either lead through the plume or involved crossing a bridge. The bridges could at the time these were crossed either be open or closed. The two routes generated using the two algorithms are depicted in figure 67.

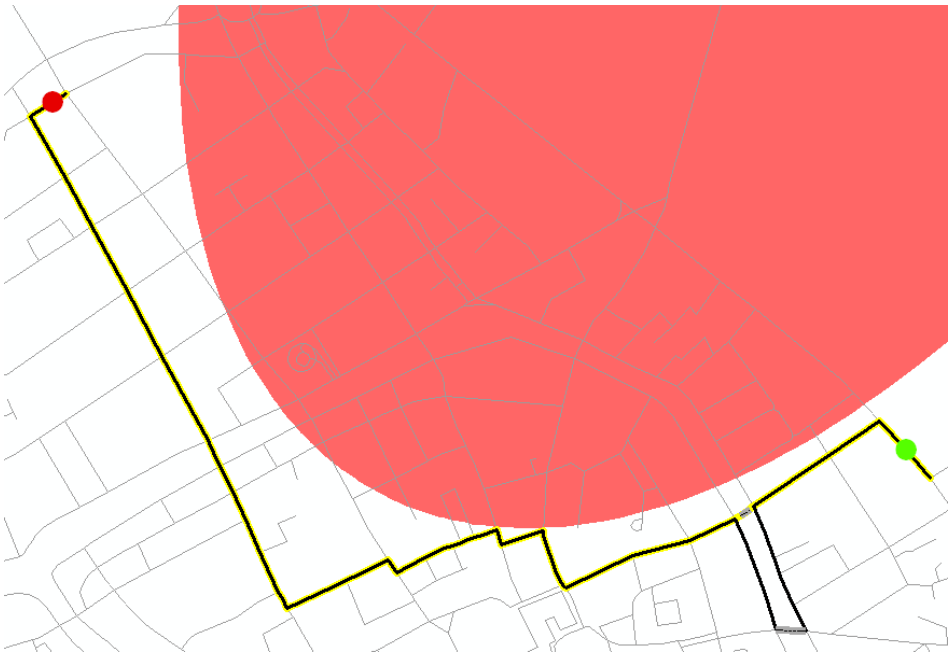


Figure 67: Route considering the bridge (black) versus normal route(yellow)

As can be seen here the routes in this case are also relatively similar. A close up of the difference between the two routes is depicted in figure 68. Apparently here it pays off to move around the first bridge and take the second bridge over the water. This is caused by the disadvantageous opening times of the upper bridge in figure 68.

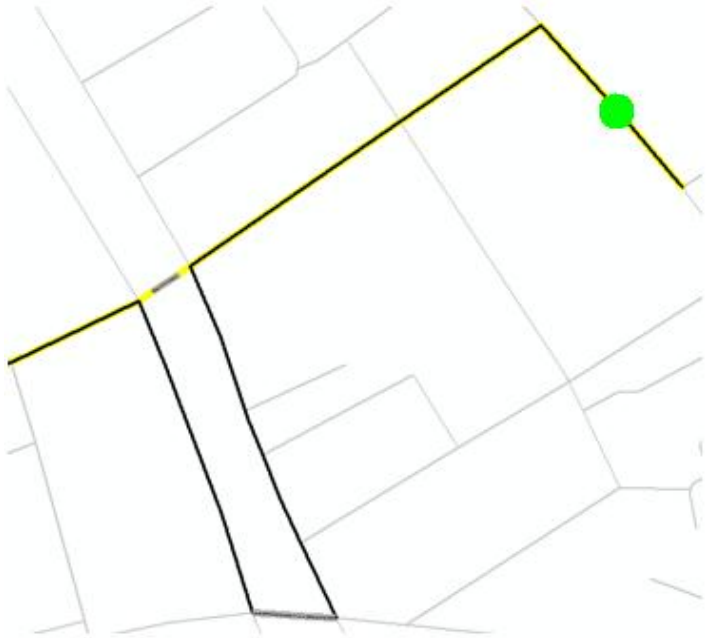


Figure 68: Alternative route around the closed bridge

To represent the difference in actual travel times a schematic representation similar to the representation produced for the previous case is depicted in figure 69.

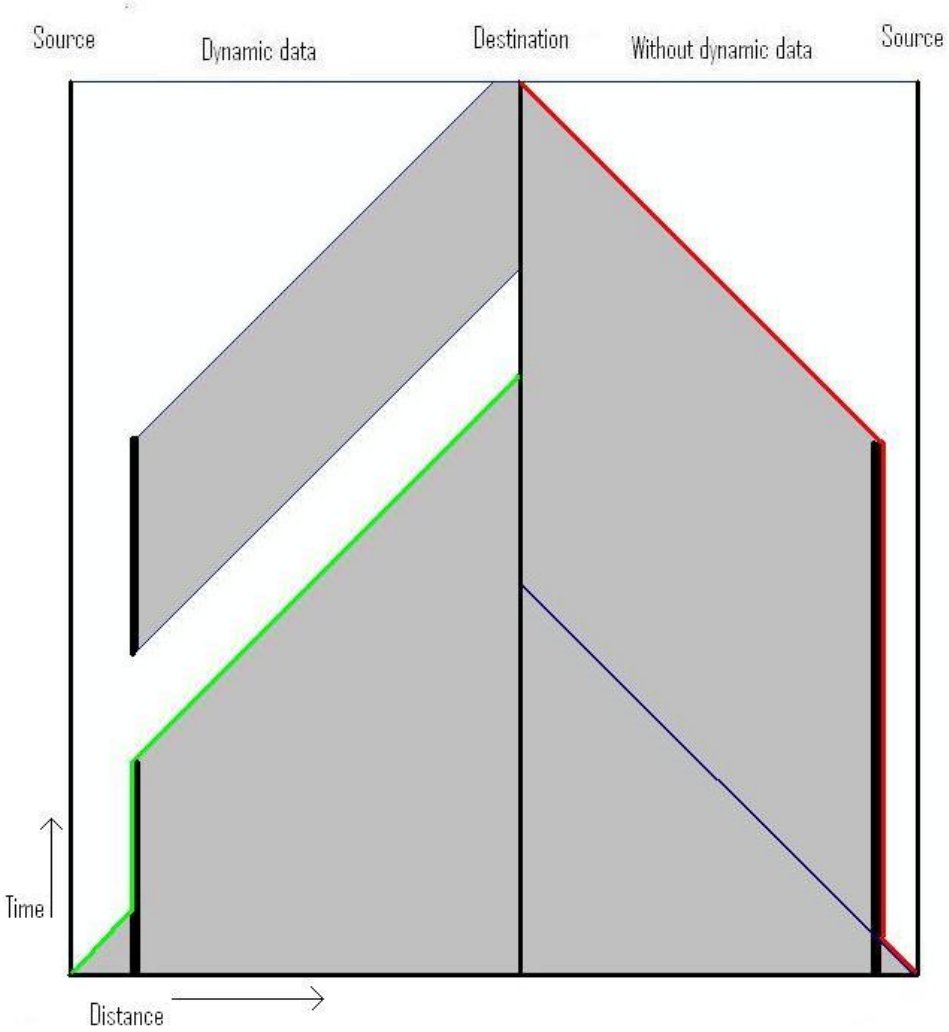


Figure 69: Schematic representation of the two routes

In this case the difference in travel times is larger as can also be deducted from table 16.

Table 16: Difference in anticipated travel time and actual travel time (Case 2)			
	Anticipated travel time	Waiting time	Actual travel time
Route considering predictions (black)	5,594	1,389 (expected)	5,594
Route without considering predictions (Yellow)	3,718	4,640 (unexpected)	8,358

In this case the advantage of incorporating the route becomes clear. The saving in time that can be achieved by using the bridge times is considerable. The large period the bridge is open for triggers this difference. This means that using the algorithm that does not consider the dynamic data, a route is produced that in this case is almost three minutes longer than necessary.

8.2.3 Case 3: Delft

In the case of Delft the two algorithms produce very different roués, which is depicted in figure 70. The route considering the bridges moves through the city center, where the route not considering the dynamic data moves around the city center.

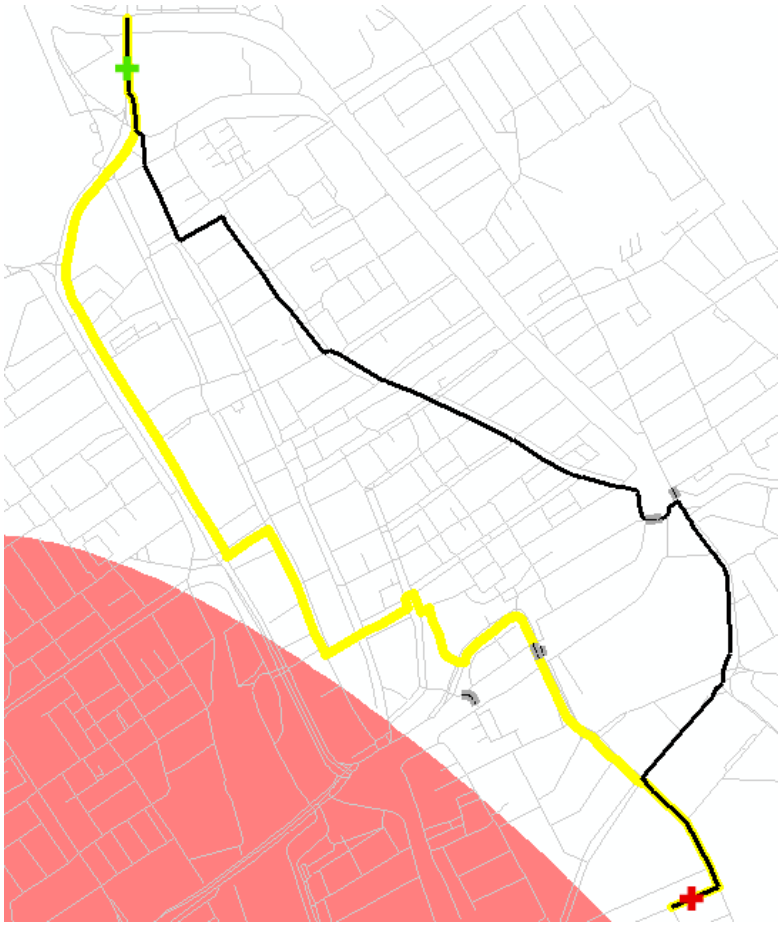


Figure 70: Route considering the bridge (black) versus normal route(yellow)

The difference in travel times are depicted in a schematic overview of the two routes in figure 71 as well as in table 17.

Table 17: Difference in anticipated travel time and actual travel time (Case 3)

	Anticipated travel time	Waiting time	Actual travel time
Route considering predictions (black)	4,548	0,129 (expected)	4,548
Route without considering predictions (Yellow)	3,796	1,158 (unexpected)	4,954

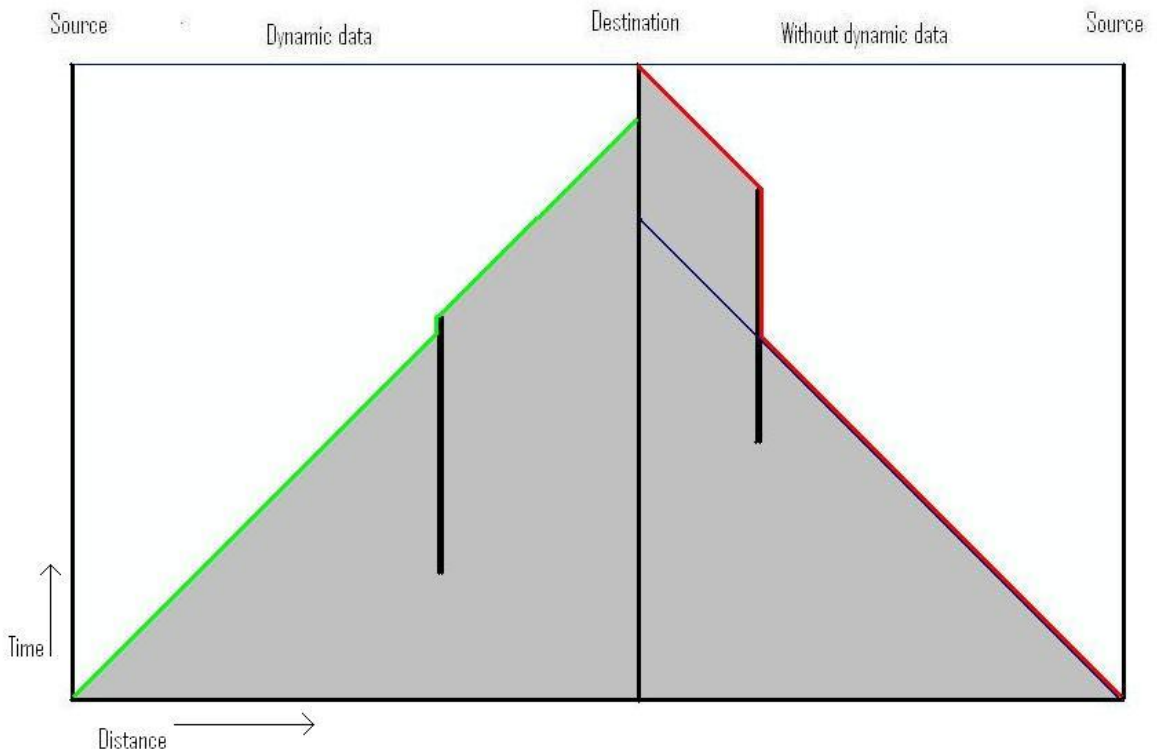


Figure 71: Schematic representation of the two routes

In this case the routes generated are clearly different from each other. The difference in total travel time is relatively small, but the route determined using the algorithm incorporating the dynamic data delivered the quicker route and a better estimation of the actual travel time.

8.2.4 Case 4: Moving plume

In all the previous cases the dynamic data consisted of bridges that were closed for traffic. In this case the dynamic data represents a moving plume. The two routes the different algorithms generated are depicted in figure 72.

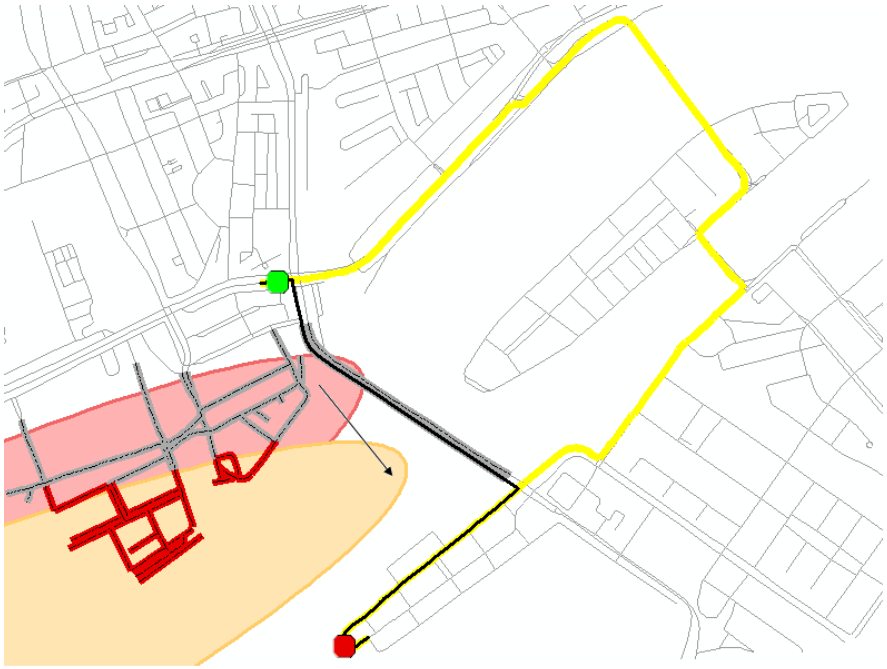


Figure 72: Route considering the bridge (black) versus normal route(yellow)

In this case the algorithm considering the roads closed as a result of the predicted plume movement delivers a shorter path looking at the distance covered. The actual travel times of the route are depicted in the schematic representation (figure 73) and in table 18.

Table 18: Difference in anticipated travel time and actual travel time (Case 4)

	Anticipated travel time	Waiting time	Actual travel time
Route considering predictions (black)	4,733	2,358 (expected)	4,733
Route without considering predictions (Yellow)	4,958	0,000 (unexpected)	4,958

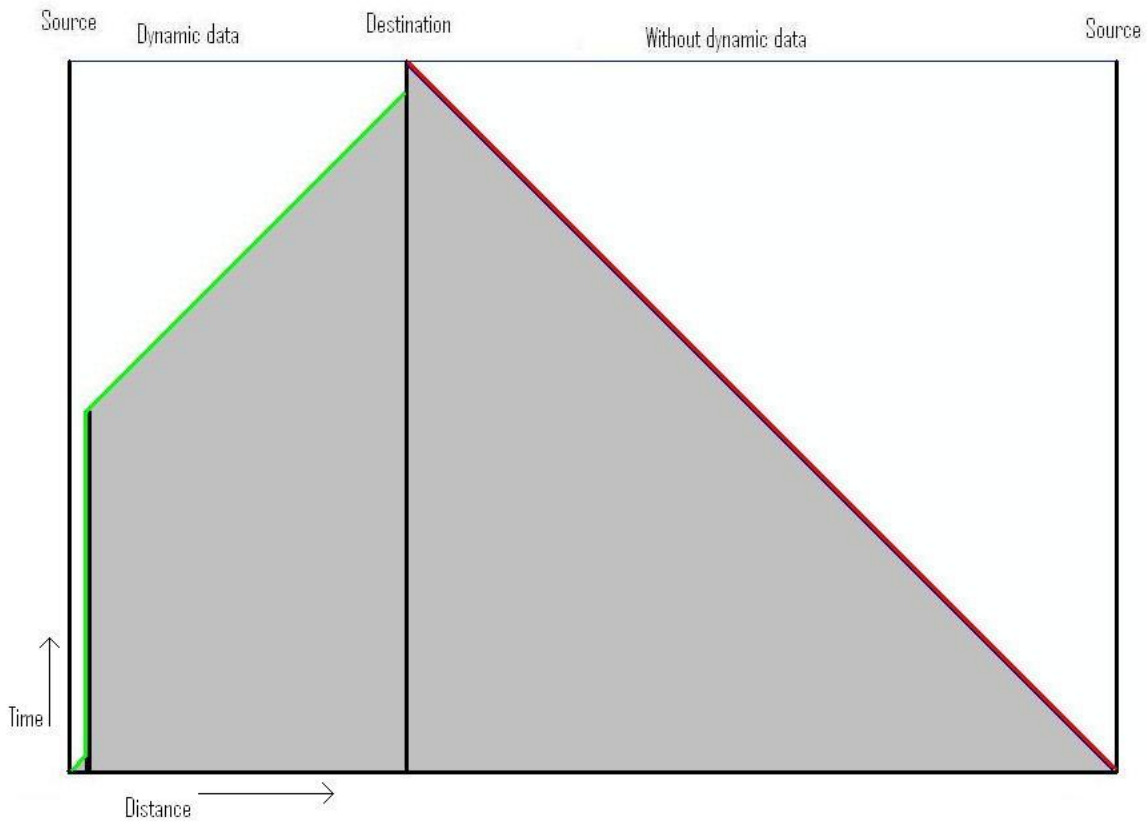


Figure 73: Schematic representation of the two routes

In this case the route determined without considering the dynamic attribute of the plume predicted the correct travel time. This is caused by the fact that the route moves around the affected roads and does not encounter any bridges on its way. The important thing here however is, that waiting for the plume to move away results in a shorter travel time than moving around the plume.

8.2.5 Case 5: Destination within the plume

In this final case the destination is inside the plume. As can be seen in figure 74 there are a number of bridges between the source and the destination. The two routes generated are in this case rather different.

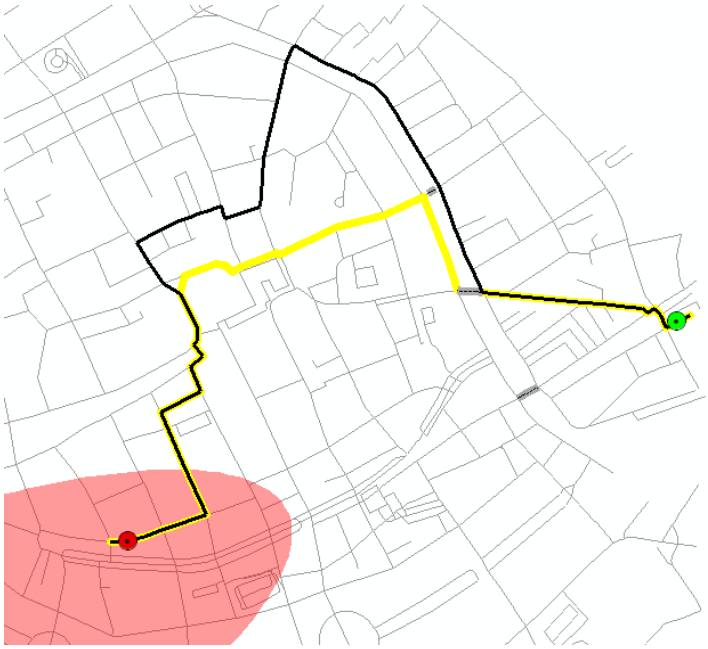


Figure 74: Route considering the bridge (black) versus normal route(yellow)

The route considering the plume in this case seems longer than the route not considering the dynamic data. In table 19 however the opposite result is represented, due to the unexpected waiting time. The route depicted in yellow includes a bridge that is closed for traffic when the vehicle is supposed to arrive at that point. This bridge however was not incorporated into the algorithm. The route that did consider the bridges is depicted in black and moves around the bridges. In figure 75 the two routes are depicted schematically.

Table 19: Difference in anticipated travel time and actual travel time (Case 5)

	Anticipated travel time	Waiting time	Actual travel time
Route considering predictions (black)	3,722	0,000 (expected)	3,722
Route without considering predictions (Yellow)	3,389	1,462 (unexpected)	4,851

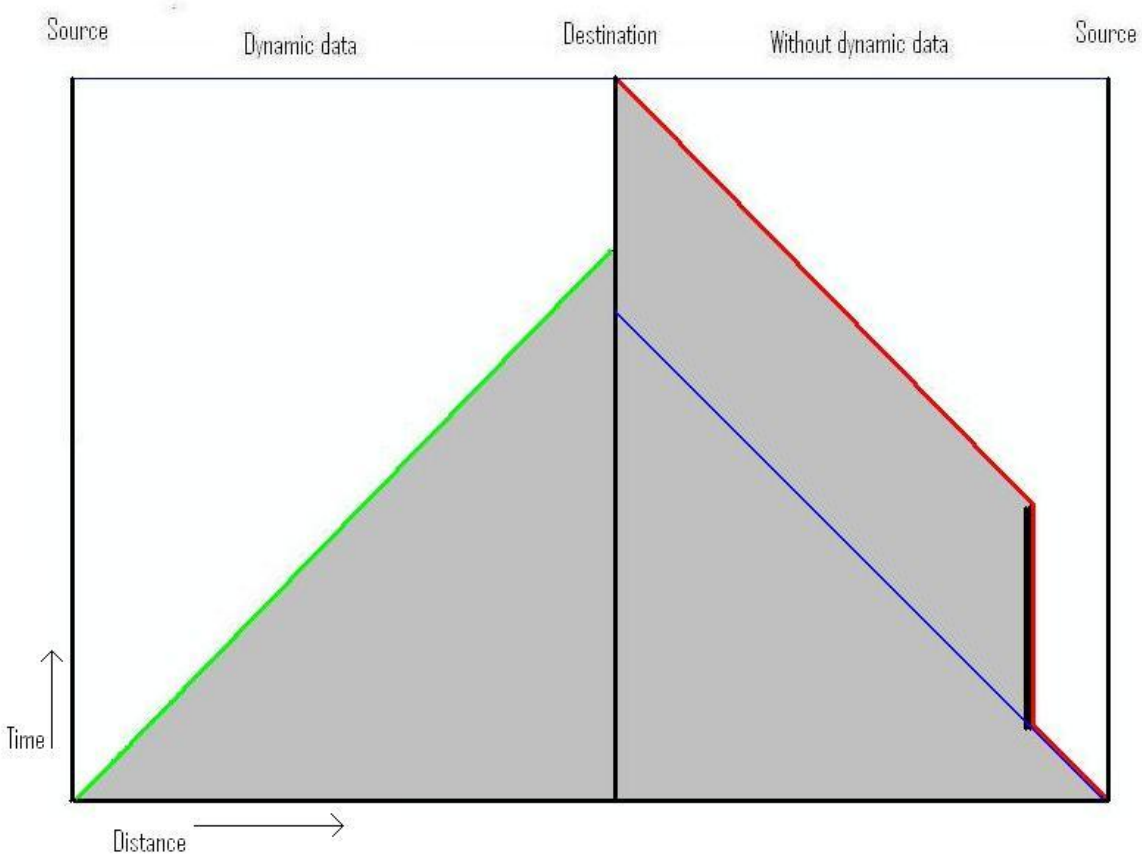


Figure 75: Schematic representation of the two routes

Also in this case the route considering the bridges delivers the best estimate of the actual travel time.

9. Discussion

9.1 Introduction

Throughout this report a couple of assumption have been made that helped to narrow down the research area. In addition these assumptions helped to focus not as much on the resulting route, but on one of the key processes in determining that route; the algorithm. The other elements in the navigation process were not addressed in as much detail as their importance for the result of this process might deem necessary. Therefore the elements that were assumed to be constant in this research are discussed here as well as the research subject itself.

9.2 Information

The first element that was assumed to be complete, up-to-date and available is the information about travel times, restricted areas, road availability, bridge opening and closing times, the source and the destination of the route. Though the latter two are vital for the determination of a route, the others have implications for the reliability and quality of the route determined. When a source or a destination of the travel are missing the route cannot be determined. However when one of the other elements are missing a route is determined, but does not consider all the aspects the user expects it to consider. This might lead to surprises. The route the algorithm produces might lead the emergency unit directly towards a traffic jam not knowing it exists. The same goes for bridges that are closed for traffic and roads that are to damaged to use. Because the case of a disaster area is characterized by a relative chaotic environment with rapidly changing situations on the road and around the road, the chance that some of the data is inaccurate, incomplete or simply outdated is bigger than it is in normal situations. Below the different information types are discussed separately.

9.2.1 Source and destination

The information on the source of the emergency response unit is could be a GPS location, when in vehicle navigation is used. The problem with GPS in built up areas is that the quality of the signal can deteriorate when located between high-rise buildings. Other than that the GPS technology is quite reliable and commonly used for navigation purposes. The destination however is assumed to be delivered to the emergency response unit in the field by the emergency room. This introduces a necessary transfer of data through a wireless network. In case of a disaster the wireless network such as the GSM network might be damaged or overused. The delivery of the information is at risk in these situations. Without the destination as input to the process the gathering of all the other information is pointless, since no route can be determined without a source and a destination. In cases like this the destination could be communicated through different mediums. An example of a separate communication network for emergency services is C2000. C2000 allows emergency services to communicate verbally with each other even when other networks are down (vstPN, 2009).

9.2.2 Travel costs

In the calculation of the route the actual travel costs are important to make sure that the route found is also in reality the optimal route from the source to the destination. The information on the current traffic conditions and the travel times that are a result of these conditions are estimations of the actual situation. The number and quality of these estimations depend on the

availability and characteristics of the methods of measurement. When the information is of a high quality and covers the complete network it is however still information on the current situation. By the time the emergency unit is half way along the route the travel costs are likely to have changed a couple of times already. The incorporation of predictions for the travel costs could help to improve the reliability of the estimated travel costs of the shortest path. In case of a disaster it is however unlikely that the predictions for a normal day still apply. Either the models to predict the traffic conditions should be able to adapt to such a specific situation or the traffic information should be updated constantly. The latter asks for the algorithm to be implemented into a process which takes care of the re-evaluation of the route. The incorporation of predictions into the algorithm can be done similar to the implementation chosen for the bridges in this research.
The hard thing in this research is that the travel costs are a combination of actual travel times that go for most of the network and a travel costs that represents the factor danger. This applies only to the part of the network that is covered by the restricted area. This could be a toxic plume, a flooded area or something like a forest fire. Though the aim of these costs is to make sure that the route will not cross this area, the estimated time of arrival will be higher for cases where the destination is inside this area, due to the increased travel costs. Another issue here is that the choice of the multiplication factor influences the chance that the shortest route will be found through the restricted area. If the multiplication factor is 2, meaning that all the roads covered by the restricted area have their travel time multiplied with 2, the chance the shortest route crosses the restricted area will be significantly larger than in cases where the multiplication factor is set to 100. Based on the kind of toxics, the concentration of these toxins and user needs identified the choice of a factor is a balancing act.

9.2.3 Road availability

The quality of the road data is the first important factor that influences the quality and the completeness of the route determination process. The base availability of the roads can be defined by the existence of roads in the network data or be specified further by the attributes. One can imagine that different types are able to drive on different types of road. A big truck for example might not be able to drive on small dirt roads through a forest where a small 4x4 vehicle is able to do so. Apart from the weight or power of a vehicle also the width, length and height of the vehicle can influence the availability of roads for that vehicle. Examples of restrictions can be steep slopes, sharp turns or width of bridges. Apart from these considerations that focus mainly on the vehicles driving the road, the road itself can become unavailable due to construction work or damages. The latter of the two not only asks for constant updating but also for detection. Meaning that the locations the road is damaged are only known once at least one vehicle tried to travel along that road and reported the damages. This information can be reported and used in the next route determination process. The vehicles however that are already on their way need the route to be updated with this new information. However the choices these units already made while navigating to their destination might cause the updated route to be far less optimal than the second best option from their original starting point. Without any doubt this information asks for wireless communication between the different navigation devices. Though the availability of the wireless communication network to all the vehicles in the area and to the central emergency room can be a bottle neck in this process another problem is the processing of the new information. The information needs to be sent quickly after acquisition and needs to be accurate with respect to its nature, severity, duration and location. Each of these factors influences the value and use of the information in the process of routing. If the location is not known for example it will be pointless to incorporate it in the process of routing. If the nature

and severity of the damages are unknown it is impossible to determine whether the damages will block the road completely or that the road is only blocked for vehicles more than 2 meters wide or vehicles without 4x4 drive. The duration of the blocking might prove to be valuable just like the information about the moments the bridges are closed for traffic. It may in these cases become an option to wait for the road to be cleared again instead of taking an alternative route.
In this case one of the elements that makes roads unavailable to traffic is the restricted area. In this research it has been an assumption that this restricted area represents for example a gas plume, a flooded area or a forest fire. The area represented on the map is assumed to be the only part of the network that is affected by this particular event. All the roads within that restricted area are equally affected by it. This is an assumption that works well in analyzing this problem but in reality it will be hard to capture the event in a single restricted area. Basically the considerations for any kind of restricted area are the same. The danger of the event is not equally spread throughout the restricted area. The concentration of toxins is likely to vary within the plume and the travel costs should respect these different densities. The predictions of the plume or restricted area are estimations of the development of the plume. Whether these estimations turn out to be right cannot be said beforehand. It is however likely that the estimation is not exactly correct. This means that the actual location of the restricted area as it was predicted should be checked against the actual plume. The danger might be more or less severe based on the nature of the direct environment, but also simply because of the distance to the origin of the danger.

9.3 Route determination

All the information is processed and the result of the calculations performed on the data is a route that can be followed by the person driving a vehicle. This route is the fastest route to a destination based on the available information. If the information changes the route can be determined based on the new information. As long as the information is available and usable in the process of determining the route no problems might be anticipated. There is however a problem that might in some cases become very obvious to the driver. This problem relates to the process of updating. In the process of updating new information is used to update the existing route according to the current situation. The original route however does not anticipate this kind of changes and the road that results from the updating process doesn't either. It is possible to consider the possible alternatives to a route in determining the best route. It might be that the fastest route does not have any alternatives close to it. If the route becomes blocked this results in a large amount of extra travel distance. The route with good alternatives might not be the fastest, but the extra travel time associated with changing the route might be far less. This approach to the problem is more suited for cases where updating is the most important feature in the research. In this case however the focus was more on the (temporal) closing of roads. This means that the results after updating can become less optimal than the results of another algorithm would have been in the same situation. When comparing the algorithm produced here in relation to the algorithms used currently for navigation purposes the latter are more refined. Examples of these attributes are the restricted turns, one way traffic and permitted traffic types. In the algorithm produced one way roads are handled as if they were two way roads and restricted turns are completely ignored. In addition it is assumed that all the roads in the network set are available for the emergency unit, who is the intended end user of the algorithm. Another design choice has been to generate a new graph for every new request for a fastest route. This is a relatively time consuming process, but was deemed necessary to be able to change the travel times and availability of every road segment. For improving the performance the data structure of the road set could have been

improved. This would not have changed the actual result of the algorithm, but these would then be produced a lot faster.

9.4 User response

The overall success of any result of the routing process, depends on the response the user has to the route provided to him. If the information is displayed ambiguous or incomplete the instructions cannot be followed by the driver. In this case the calculation of the original route and any route in the future is pointless unless the information is displayed properly. Another point of interest here is that the timing of providing the information is also important. Is the directional information provided to early than the driver is likely to have forgotten about it, by the he needs to make the turn. Though this seems to be a completely different element of the process of navigating, the information that can be provided to the user is partly delivered by the algorithm. One should think of things like the estimated time of arrival, the total waiting time, the distance travelled and the route itself. If this information is too complex the user will not be able to respond to it.
A challenge in the communication is to convince someone who is in a hurry that it is faster to wait until the bridge is open for traffic again instead of taking another route. Waiting means that you're not moving and that is not perceived as progress and thus undesirable for someone who is in a hurry. In this case however one would only be asked to wait if it is faster than any other alternative route. The navigation device however does not have the ability to take over control of the vehicle and thus the decision of the driver determines whether the route followed corresponds to the route suggested by the routing algorithm.

9.5 Event handling

Throughout the development of the algorithm little attention was given to the iteration and re-evaluation of the route. As mentioned earlier in the research based on Montemanni (2002), Larsen (2001) and Golshani et al (1996), the characteristics of the environment the algorithm is implemented in decide what type of re-evaluation is the optimum. The main choice was between the trigger of a re-evaluation of the route. The first option was to base the re-evaluation on the occurrence of changes. The second option was to have a time interval, meaning that the route will be re-evaluated every x minutes. The first one is an interesting one since the amount of changes depend apart from the degree of chaos in the environment on a number of things. Amongst these are the total area changes are recorded in, the percentage of the changes that are recorded and the number of changes send simultaneously. What is important to consider is that the amount of changes that lead to a re-evaluation is influenced by the incorporation of predictions. If information on bridges opening and closing is incorporated as predictions into the algorithm, the event of a bridge closing for traffic does not have to be sent to the emergency unit in the field as a change. This event however does add to the amount of changes that are occurring in the environment. So a chaotic environment asks for a dynamic approach to re-evaluation. When the majority of the changes can however be predicted within this chaotic environment, this environment is not chaotic for the algorithm. The approach to re-evaluation that suffices to deliver a good result might in this case be semi-static approach. By incorporating predictions in the algorithm the degree of dynamism in an environment should be re-evaluated using an estimation of the percentage of changes that are not predicted for this environment. Based on these estimations an approach to re-evaluation should be determined. A second challenge here would then be to filter the changes in the environment for changes that were predicted.

9.6 The Algorithm

For the determination of the route introducing the possibility of waiting somewhere along the route the Dijkstra algorithm was implemented and altered to allow for waiting. A drawback of the algorithm as implemented here is that the performance decreases when the number of roads increases. According to the tests performed this is not due to the introduction of the possibility to wait. All the roads that are explored but not yet assessed for their total travel costs are stored in one list. This list grows every round and becomes clogged. As an alternative it was tried to order the list, but the problem persisted. A solution would have been to implement a priority queue. Due to the fact that this algorithm was developed to explore the possibilities of introducing waiting time in the determination of the route, this is not such an important limitation as it would be in practice. The goal of the algorithm to explore the possibility to introduce waiting times has also implications for the user friendliness of the algorithm. First of all the source and the destination are stored and read from a simple shapefile.

Currently the times the bridge is closed or open again are stored as an amount of time relative to the starting time of the algorithm. For one bridge the time it closes could for example be denoted as 5. This means that the bridge is closed from the fifth minute of the trip until the end time which could for example be set to 10. This means that depending on the time of arrival at the bridge the unit has to wait up to a maximum of 5 minutes. In practice this schedule, if available, will be expressed in clock times. The time of arrival at the bridge was assumed to be the same as the time of arrival at the start of the road segment. Based on an exploration of known bridges in the Multinet data this assumption seems to hold.

The route that is generated by the algorithm starts with the line the source is on. The travel time for this line is set to zero. This was done to make sure that a trip from source to source gets a travel time that is equal to zero. For the destination also the entire road segment is set as the destination line. The travel time for this road segment is always completely added to the total travel time of the trip even if the destination is at the start of this road segment. The route is therefore longer than the actual trip. The travel time however does not correspond to the length of the route drawn, because the travel time for the first segment is not considered. These problems relate to the fact that in this implementation of the algorithm points are the elements that connect the lines. Normally lines would be considered the elements that connect the points. When the lines are connecting the points, the point is where the algorithm searches for other points that are connected with it through a line. In this case the algorithm looks which lines are connected to the current line in a point.

The dilemma here was that the source and the destination of the route could be half way along a road segment. If, in this implementation, lines were considered to be connecting the points a new point should have been created that exactly coincides with the source or the destination of the route to be determined. This means that the road segment the source or destination is on needs to be split in two new segments. For both these segments the “from” and the “to” node have to be determined again as well as the new travel time. For the algorithm it is not really important whether the lines are connecting the points or that the points are connecting the lines. The choice made in the design of the algorithm only shows in the resulting route. This resulting route should be cut at the destination point and at the source point, keeping the route line that visually connects the source and destination. The procedure needed to do this is part of the process that follows the route determination process, but adds to the (visual) quality of the route determined and therefore influences its success. Another output provided by many navigation devices is the Estimated Time of Arrival (ETA). This is similarly to the opening and closing times of the bridges a time relative to the starting time of the trip. So the ETA, normally the start time plus the travel time plus the waiting time, is not as much an ETA as it is an estimation of the total travel costs including the waiting time. Another important piece of

information that is not currently provided by the algorithm is the Total Waiting Time (TWT). This information can be retrieved, but this is not part of the current implementation. The indication of the TWT might add to the success of the algorithm and the navigation process as a whole, since it gives the user an idea of what is going to happen. In addition it reassures the user that the algorithm took this period of waiting into account while determining the quickest route.

10. Conclusion

10.1 Answering the research questions

Throughout this research attention has focused on the algorithm used to determine the route. It was shown that by altering the algorithm waiting could be incorporated in the route determination process. The majority of the research aimed to develop a way to introduce the possibility to wait into the algorithm. This was achieved by calculating the difference between the time of arrival at a block and the time the block was opening again. Using this method it was shown that not only temporal blocks like bridges, but also constant blocks and moving plumes could be incorporated. Through a number of cases the differences between the algorithm that allowed for waiting and the normal algorithm was shown. In all these cases the results of the algorithm allowing for waiting time delivered the same or better results than the normal algorithm.

The plume played an important role during the kickoff of this research, as an example of changing information that can be predicted. The main focus for the route was to move into the plume downwind. The assumption made was that the destination of a travel inside the plume was the source of the cloud. This means that the downwind approach would be one of the shortest paths into the plume. After combining this notion with the user need to stay inside the plume as short as possible and the remark that it was even better not to enter the plume at all, the usage of a constant penalty factor became in sight. Using this constant penalty factor, by multiplying it with the travel time of the road segments underneath the plume the shortest path into the plume became the most desirable. Making the assumption that the only destination in the plume would be the source of the plume, this shortest path implicated a downwind approach. This downwind approach however is not implemented by using the direction of the roads, but only by the shape and location of the plume. The choice of a value for this penalty factor turned out to be a balancing act between the discomfort of travelling through the plume against the discomfort of travelling additional time around the plume.

The incorporation of the additional information on road availability and the temporal blocks could be done in two different ways. The first is to change the travel costs of a road segment. This option is most useful for situations were traveling over a road segment is possible at any time. The time of arrival however in this case does not make any difference for the costs of travelling over that road segment. This additional travel time for a road segment was used to represent additional risk or discomfort. This additional travel time was caused by travelling through a static plume or opening a dynamic cone for example. The second option is to introduce a waiting time. This method was used here in cases that a travel was not possible on a particular edge. Examples of roads that should have a waiting time associated with them are train crossings, bridges, moving plumes and constant blocks. A constant block could be a road construction or road not suitable for the vehicle that is currently used.

Though in this case the schedules of the bridge open and closing times were known before hand, these predictions in practice might be unavailable. In this case it becomes important to acquire the information needed and re-evaluate the route determined when needed. Some examples were found where current information was used as a prediction for the next vehicle to be routed. This means that information on network speed half way through the network was considered in calculating a new route from the source to the destination. A downside of this approach is that by the time the vehicle routed is at the location of the delay, the information

on that delay is already outdated. This means that the delay could have become worse in the meantime or that the delay has completely vanished. Reason for this approach however is that the predictions on travel times in the future can vary widely in quality. The predictions could be improved by developing more advanced modeling techniques or by using data collected on days with similar circumstances. This not only touches upon the key requisite but also on the need for re-evaluation. The key requisite for the algorithm is the validity of the predictions used. If the predictions cover every aspect of the travel exactly, there is no need to re-evaluate a route. Since it is impossible to predict every accident or movement re-evaluations will always be needed in the process of routing. Though the choice of re-evaluation method has not been implemented, the best anticipated method of re-evaluation was determined to be a mix between the event re-evaluation and the interval re-evaluation. This means that a re-evaluation takes place after a number of changes to the input data. If there are however too little changes to the input data the re-evaluation should at least take place every time interval, for example every five minutes. In this way the resulting route will be as up-to-date as possible in any environment, both the extremely chaotic as the extremely stable.

This leaves the question whether a disaster situation is really a chaotic one, looking at the changes to the network. Though it is out of scope for this research a division can be made between the chaos in such a situation within the input data and on the input data. Meaning that it might be anticipated that except for some major changes to the network at the start of the disaster, the network will remain relatively constant during the disaster. The main chaos will however take place on the network. Here a combination of life threatening circumstances, panic and uncertainty will ask for all the attention of the emergency response unit.

For the emergency response units all the information can ideally be made available. It is however not realistic to expect these people to combine all the information and make informed decisions based on that information under a lot of stress. Under stressful conditions, decisions that are not related to the actual work of the emergency response unit should be taken away from them. As was shown in this research, the routing decisions are an example of this kind of decision making. It has been elaborated on that navigating consists of a large number of processes that each need to be implemented successfully to make the navigation process a success. In this research a useful addition might have been made to automatically consider dynamic data into the route determination process, but it is the way the input data is gathered and the resulting route is presented that are decisive for the success of the route.

The adaptation of the algorithm is aimed as a prove of concept, simply to show that dynamic data can be considered in a route. The input predictions used however were all fictional. The research has shown that the safety can be improved by introducing the plume in a static Dijkstra algorithm. In chapter 7 and 8 it was shown that especially the unexpected waiting times, resulting in unexpected delays, can be countered by using predictions. The total travel time estimated for the route is therefore likely to be more accurate when the adapted algorithm is used. This adaptation of the algorithm allows one also to use predictions of the plume. This not only makes sure that the route is safe, but will also suggest one to wait for the plume to wear off. This is only done if waiting results in a smaller travel time than taking an alternative. So the adapted algorithm not only produces safer routes, but also compares the waiting times with the additional travel time of taking an alternative route. The incorporation of the predictions into the algorithm proved, during the execution of the cases, not to have a large negative impact on the performance of the route computation.

In the next section attention will focus on the elements of the route determination process that deserve additional attention.

10.2 Future work

The improvement of the routing results is an ongoing process of identifying weaknesses, performing research and improving the processes. For the incorporation of dynamic data the algorithm has been adapted, but some improvements can still be made to for example the input data of the algorithm.

10.2.1 The plume

To get the best overview of the health risk of the plume information is needed on the concentrations of the toxics inside the plume. The plume is currently a polygon with a constant value. In practice however the concentrations will differ across the plume, having a higher concentration close to the source or centre of the plume and a lower concentration on the sides. In addition the plume is considered here as a 2D object blocking all the streets that cross it. A real plume of toxics is a 3D object not only having a length and a width but also a height. There is a good chance that the largest portion of the plume, however high the concentrations of toxics, does not impose any risk on the people on the ground. This because the plume is simply higher up in the air. It would in this case be possible to navigate underneath the plume without any problem. The 3D aspect of the plume can also be extended onto the network. This means that moving through a tunnel is still safe even when the plume does touch the surface of the earth. Using the same line of reasoning an elevated road could still be obstructed by the plume, while all the other roads around it are safe to drive on.

10.2.2 Travel time assignment

For the roads underneath the plume the travel costs are altered by multiplying them with a penalty factor. This is done for all the roads that intersect the plume. This means that a road which is underneath the plume for 10 % of its own length is punished as hard as a road segment that is underneath the plume completely. In addition these penalty factors should depend on the concentrations of toxics the people driving on that road are exposed to. Since there is a desire to move downwind towards a destination close to or inside the plume, this downwind driving should also be included in the travel costs. This involves not only the calculation of the orientation of the road segment with respect to the wind direction, but also a detailed mapping of the wind directions. Especially within urban areas the layout of the built up area, the height of the buildings, their orientation and the width of the space between them influences the wind direction speed. When one starts to differentiate between the wind directions, the next step will be to include the wind speeds. Additional research is then needed on the ways to translate models of wind directions in built-up areas, models of spreading toxics and the combinations between them into travel times of a road segment.

10.2.3 Travel time acquisition

For the correctness of the input data, describing the future situation (predictions) and the current situation attention needs to be focussed on the methods of acquisition. An important element of input data for the travel times is the road length, the maximum speed, but even more important the measure of congestion. To really deliver a good prediction of the travel

time of a road segment by the time that an emergency vehicle arrives at that road, the early signs of upcoming or disappearing congestion should be recognized. But even if the congestion is known exactly, still the travel times can turn out to be different in reality. The predictions of the travel times would really benefit from knowledge about the response of people to the emergency unit. This response is influenced by their ability to manoeuvre, the layout of the road, the amount of time to respond to the advancing emergency unit and their experience in dealing with such situations. It could be that congestion on one road would be less of an obstacle for an emergency unit to travel through than a road that is not that congested. This could be the case for a 3 lane road against a 1 lane road for example. This not only provides an additional challenge for the models used for predicting, but also opens the door to ideas about influencing the travel times.

10.2.4 Influencing travel times

When knowledge has been gathered on ideal situations for the emergency unit to move through, methods can be thought of to create the ideal situations. This could for example be done by influencing the navigational devices of others or influencing signs above the highways. The communication between the navigation device of the emergency unit and the navigational devices of all the other vehicles in the area could help to clear the way for the emergency unit. One of the options would be for the navigational devices of some vehicles to take an alternative route, to prevent congestion and create additional moving space for the emergency unit. A second option to create additional moving space for the emergency response units is to communicate with the signs above the highways. Based on the routes of these emergency response units one of the lanes could be closed for all traffic except for the emergency units. This not only provides a fast connection for the emergency response units that were already routed over that highway, but also for other emergency units. The closure of one of the lanes could mean for a number of other emergency units that a faster alternative came into existence.
Apart from clearing the way for the emergency units such a form of communication could also help to coordinate the evacuation effort. If it is possible to communicate the existence and location of dangerous zone to all the vehicles in the area affected, some central evacuation routes could be established. In this way the evacuation can be guided, providing additional benefits in terms of lower travel times for the emergency units. The important thing here is that the communication lines should be robust in case of disaster and not subject to overburden.

10.3 Possible applications outside disaster situations

Though the focus has been on the disaster area including toxics, flooding or forest fires, the same principle could be used elsewhere. One can think about warzones, but also situations as simple as unsafe zones in rural or urban areas. These zones should not be crossed in times of heavy fighting or for example during the night. In addition the simple desire to implement the same kind of algorithm in more situations will help to identify and solve challenges faster, adding to the quality of the resulting route in all situations.

11. Reference

- Bemmelen, J. van, W. Quak, M. van Hekken, P. van Oosterom, 1993. "Vector vs. Raster-Based Algorithms for Cross Country Movement Planning", Auto-Carto 11, Minneapolis, Minnesota, October 30 - November 1, pages 304-317.

- Black, P.E., 2005. "Bellman-Ford algorithm", in: P.E. Black (Eds), 2005. "Dictionary of Algorithms and Data Structures". U.S. National Institute of Standards and Technology. Website: http://www.itl.nist.gov/div897/sqg/dads/HTML/bellmanford.html

- Bruynooge, G. (2008). "MDT als integraal deel van de beheerorganisatie", Presentation during conference 24-04-2008 in Arnhem. Internetsite: http://www.nifv.nl/upload/128279_668_1209122868953-MDT_Gerwin_Bruynooge_-_IT-Solutions.pdf

- Cormen, T.H., C.E., Leiserson, R.L., Rivest, 1992. "Intorduction to Algorithms". MIT Press Cambridge, Massachusetts

- Di Caro, G. A., F. Ducatelle, L. M. Gambardella, 2008. "Ant Colony Optimization for Routing in Mobile Ad Hoc Networks in Urban Environments". IDSIA, may 2008. Internetsite: http://www.idsia.ch/reports?period=all

- Dijkstra, E.W., 1959. "A Note on Two Problems in Connexion with Graphs". Numerische Mathematik, Volume 1, Issue 1, December 1959, pp: 269-271. Internetsite: http://www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf

- GDI4DM, 2006. "Geospatial Data Infrastructure for Disaster Management Appendix 2 Grant Application template". Private communication with Sisi Zlatanova (TU Delft) in November 2008.

- Geurts, P., 2008. "Informatiehuishouding", Presentation during conference 24-04-2008 in Arnhem. Internetsite: http://www.nifv.nl/upload/128296_668_1209382042984-Infohuishouding_Paul_Geurts-InfoArchitect_Gemeente_Nijmegen.pdf

- Golshani, F., E. Cortes-Rello, T.H. Howell, 1996. "Dynamic route planning with uncertain information", Knowledge Based Systems, Volume 9, Issue 4, pp: 223-232. Internetsite: http://www.ingentaconnect.com/content/els/09507051/1996/00000009/00000004/art01031

- Hanshar, F., B. Ombuki-Berman, 2007. "Dynamic vehicle routing using genetic algorithms". In: Applied Intelligence, volume 27, pp: 89-99.

- Hart, P. E., N. J. Nilsson, B. Raphael, 1968. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". Systems Science and Cybernetics, IEEE Transactions on, Volume 4, Issue 2, July 1968, pp: 100 – 107.

- Kieboom, R (CTO at CityGIS), 2008. Private communication with R. Kieboom.

- Larsen, A., 2001. “The Dynamic vehicle routing problem”. Lyngby: IMM, DTU. Internetsite: http://www.dtu.dk/inst/CTT/English/Research/Research%20Groups/Logistics%20Optimisation%20Group/Research/The%20dynamic%20vehicle%20routing%20problem.aspx

- Lester, P., 2005. “A* Pathfinding for Beginners”. Internetsite: http://www.policyalmanac.org/games/aStarTutorial.htm

- Min, K., J. Kim, J. Park, 2006. "Optimal Route Determination Technology Based on Trajectory Querying Moving Object Database". In: Bressan, S., J. Küng, R. Wagner (Eds.) "Database and Expert Systems Applications", Lecture Notes in Computer Science, volume 4080, 2006. pp. 666 – 675.

- Montemanni, R., L.M. Gambardella, A.E. Rizzoli, A.V. Donati, 2002. “A new algorithm for a Dynamic Vehicle Routing Problem based on Ant Colony System.” IDSIA, November 2002. Internetsite: http://www.idsia.ch/reports?period=all

- NVBR (Nederlandse Vereniging voor Brandweerzorg en Rampenbestrijding), 2006. “Programma van Eisen en Wensen GIS 2006: Visie NVBR op een openbare orde en Veiligheid-breed (Geografisch) Informatie Systeem”. Arnhem: Drukkerij Roos en Roos.

- Puthuparampil, M., 2007. "REPORT DIJKSTRA’S ALGORITHM". Internetsite: http://www.cs.nyu.edu/courses/summer07/G22.2340-001/Presentations/Puthuparampil.pdf

- Snoeren, G (Bussiness Consultant for the public safety market at ESRI), 2009. Private communication with G. Snoeren.

- TNO, 2009. “GasMal: decisionsupportsysteem voor chemische incidenten”. Internetsite: http://www.tno.nl/content.cfm?context=markten&content=product&laag1=194&laag2=194&item_id=693

- vstPN (Voorziening tot Samenwerking Politie Nederland), 2009. "C2000: Het nieuwe digitale communicatie-netwerk voor politie, brandweer, ambulance en Koninklijke Marechaussee". Internetsite: www.c2000.nl

- Wang, Z., J. Crowcroft, 1990. “Shortest path first with emergence exits”. ACM SIGCOMM Computer Communication Review, Volume 20, Issue 4, September 1990, pp: 166-176.

- Wang, Z., J. Crowcroft, 1992. “Analysis of shortest-path routing algorithms in a dynamic network environment”. ACM SIGCOMM Computer Communication Review, Volume 22, Issue 2, April 1992, pp: 63-71.

- Xiao, B., Q. Zhuge, Z. Shao, E.M.H. Sha, 2003. "Design and Analysis of Improved Shortest Path Tree Update for Network Routing". Proc. ISCA 16th International Conference on Parallel and Distributed Computing Systems (ISCA PDCS), pp: 82-87. Internetsite: http://www.utdallas.edu/~edsha/papers/bin/ISCA03.pdf

Appendix I

```
###########
Created using python 2.5.1 and ArcGIS 9.3
Title: Routing in disaster areas
Subtitle: incorporating a restricted area into the route determination process
Author: Ivo Visser
Date: 06-08-2009
###########

# functions used in the algorithm

## Function that determines the line segment a source or destination point is on and returns
## the ID of that line segment

def determineLine(UnitID, point):
    gp.MakeFeatureLayer_management(point, "sourcepoint", "Id = " + str(UnitID))
    gp.MakeFeatureLayer_management(roads, "linesLyr")
    gp.SelectLayerByLocation_management("linesLyr", "INTERSECT", "sourcepoint")
    scur = gp.searchcursor("linesLyr")
    lineList = []
    row = scur.next()
    while row:
        lineList.append(int(row.getValue("ID")))
        row = scur.next()
    lineID = lineList[0]
    return lineID

## Add all the road segments in a data file to a Graph dictionary storing the ID of the
## segment, the from- and the to-node and the travel costs

def createRoadDictionary(roads):
    scur = gp.searchcursor(roads)
    srow = scur.next()
    G = {}
    while srow:
        G[srow.getvalue("ID")] = [srow.getvalue("F_JNCTID"),
            row.getvalue("T_JNCTID"), srow.getvalue("MINUTES")]
        srow = scur.next()
    return G

## Check whether the segment under consideration is already in the queue

def Contains(queue, ID):
    a = false
    for x in queue:
        if x[0] == ID:
            a = true
        else:
            a = a
```

```
    return a

## This function identifies the edges that are connected to edge just explored and returns these
## edges if they do not already exist in the queue. For the edges that are connected and
## returned the total travel cost from the source until the end of that edge is calculated

def findConnectedRoads(ID, Graph, dests, queue):
    junctions = Graph[ID]
    connected = []
    for x in Graph:
        tmplist = Graph[x]
        queueContains = Contains(queue, x)
        if tmplist[0] == junctions[0] or tmplist[1] == junctions[1] or tmplist[0] == junctions[1]
            or tmplist[1] == junctions[0]:
            if len(dests) == 0 and len(queue) == 0:
                print "error in input, source is not entered in destination set"
            elif (x != ID) and (not x in dests) and (queueContains == 0):
                listP = dests[ID]              ## list of values of the predecessor
                listC = Graph[x]               ## list of values for the segment under consideration
                travelTime = float(listP[0]) + float(listC[2])
                connected.append([x, travelTime, ID])
    return connected

## This function selects the road segment with the lowest total travel costs from the queue and
## returns that line

def MinQueue (queue):
    q = queue[0]
    minTT = q[1]
    minLine = q
    for x in queue:
        if x[1] < minTT:
            minTT = x[1]
            minLine = x
    return minLine

## This function returns a list of line ID's that are part of the shortest path by starting with the
## destination line ID and adding the predecessor of that line until the source is reached.

def ReturnRoute(source, dest, dests):
    route = []
    v = dest
    route.append(v)
    while v != source:
        v = dests[v][1]
        route.append(v)
    return route

## This function builds a SQL statement containing all the line ID's that are part of the route.
## The SQL statement is returned
```

```
def SelectRoute(path, roads):
    a = int(path[0])
    selstat = '"ID" = ' + str(a)
    path.remove(path[0])
    for x in path:
        a = int(x)
        selstat = selstat + ' OR "ID" = ' + str(a)
    return selstat

## This function calculates new travel costs for the segments that are partly within a set
## distance of the plume

def AlterTraveltimes(roads, plume):
    gp.MakeFeatureLayer(roads, "rlayer")
    gp.SelectLayerByLocation_management("rlayer", "INTERSECT", plume, "100 meter",
        "NEW_SELECTION")
    gp.CalculateField_management("rlayer", "MINUTES", "[St_Travel]* 4")

## This function calculates the original travel costs of each segment and stores it in the travel
## costs field

def ResetTraveltimes(roads):
    gp.CalculateField_management(roads, "MINUTES", "[St_Travel]")

# preparations before the start of the algorithm

import arcgisscripting, string

gp = arcgisscripting.create(9.3)

gp.overwriteoutput = "1"
gp.workspace = r"D:\Dijkstra"
roads = "[roads].shp"
sourcepoint = "Startpoints.shp"
destpoint = "Endpoints.shp"
plume = "[plume].shp"
UnitID = [unitnumber]

## set the travel costs of the roads in the network that are crossed by the restricted area
AlterTraveltimes(roads, plume)

## determine the line ID's of the lines the source and destination location are located on
source = determineLine(UnitID, sourcepoint)
dest = determineLine(UnitID, destpoint)

print "position source and destination determined"
print "collecting road segments"

## create the lists and dictionaries needed during the determination of the route
```

```
Graph = createRoadDictionary(roads)
queue = []
dests = {}

print "road segments collected"

## set initial values; the line the source is on gets a travel cost of zero and has no predecessor
ID = source
dests[ID] = [0, 0.0]

# start of algorithm

print "calculating route..."

## for the current edge the connected edges are investigated. The connected edge with the
## lowest total travel costs is set as the current edge for which the connected edges are
## identified. This process continues until the destination edge has become the current edge.

while ID != dest:
   con = findConnectedRoads(ID, Graph, dests, queue)
   for x in con:
      queue.append(x)

   min = MinQueue(queue)
   dests[min[0]] = [min[1], min[2]]
   queue.remove(min)
   ID = min[0]
   print int(ID)
print dests

## When the lowest travel cost to the destination edge has been found, the path is determined
## and the total travel cost is stored in the estimated time of arrival (ETA) variable

print "collecting results..."
ETA = dests[dest][0]
path = ReturnRoute(source, dest, dests)
print path

## A selection statement is created to make a selection in the map of all the edges that
## participate in the shortest path

selstat = SelectRoute(path, roads)
print selstat

## the travel costs that were altered to incorporate the restricted area are changed back into
## their original value to prepare for the next run of the algorithm.

ResetTraveltimes(roads)
gp.MakeFeatureLayer(roads, "Rlyr", selstat)
gp.CopyFeatures_management("Rlyr", "route.shp")
```

```
del gp
print "route determined"
print "The total travel time is " + str(ETA) + " minutes."
```

Appendix II

```
###########
Created using python 2.5.1 and ArcGIS 9.3
Title: Routing in disaster areas
Subtitle: incorporating a restricted area and waiting time into the route determination process
Author: Ivo Visser
Date: 06-08-2009
###########

# functions used in the algorithm

## Function that determines the line segment a source or destination point is on and returns
## the ID of that line segment

def determineLine(UnitID, point):
    gp.MakeFeatureLayer_management(point, "sourcepoint", "Id = " + str(UnitID))
    gp.MakeFeatureLayer_management(roads, "linesLyr")
    gp.SelectLayerByLocation_management("linesLyr", "INTERSECT", "sourcepoint")
    scur = gp.searchcursor("linesLyr")
    lineList = []
    row = scur.next()
    while row:
        lineList.append(int(row.getValue("ID")))
        row = scur.next()
    lineID = lineList[0]
    return lineID

## Add all the road segments in a data file to a Graph dictionary storing the ID of the
## segment, the from- and the to-node and the travel costs

def createRoadDictionary(roads):
    scur = gp.searchcursor(roads)
    srow = scur.next()
    G = {}
    while srow:
        G[srow.getvalue("ID")] = [srow.getvalue("F_JNCTID"),
            row.getvalue("T_JNCTID"), srow.getvalue("MINUTES")]
        srow = scur.next()
    return G

## Check whether the segment under consideration is already in the queue

def Contains(queue, ID):
    a = false
    for x in queue:
        if x[0] == ID:
            a = true
        else:
            a = a
```

```
    return a

## This function determines whether the edge under consideration is a bridge. For edges that
## are a bridge this function checks whether the bridge is closed at that particular moment in
## time and returns the waiting time associated with travelling along that edge.

def waitTimeBridge(ID, time):
    waitingtime = 0
    t = float(time)
    ident = int(ID)
    bridges = open(r"D:\Dijkstra\bridges.txt")
    while 1:
        line = bridges.readline()
        brdg = line.split(",")
        if brdg[0] != "":
            if  int(brdg[1]) == ident and t > float(brdg[2]) and t < float(brdg[3]):
                waitingtime = float(brdg[3]) - t
            else:
                waitingtime = waitingtime
        if not line:
            break
    bridges.close()
    return waitingtime

## This function identifies the edges that are connected to edge just explored and returns these
## edges if they do not already exist in the queue. For the edges that are connected and
## returned the total travel cost from the source until the end of that edge is calculated
## including waiting time

def findConnectedRoads(ID, Graph, dests, queue):
    junctions = Graph[ID]
    connected = []
    for x in Graph:
        tmplist = Graph[x]
        queueContains = Contains(queue, x)
        if tmplist[0] == junctions[0] or tmplist[1] == junctions[1] or tmplist[0] == junctions[1]
            or tmplist[1] == junctions[0]:
            if len(dests) == 0 and len(queue) == 0:
                print "error in input, source is not entered in destination set"
            elif (x != ID) and (not x in dests) and (queueContains == 0):
                listP = dests[ID]               ## list of values of the predecessor
                listC = Graph[x]                ## list of values for the segment under consideration
                wait = waitTimeBridge(int(x), listP[0]) ## waiting time associated with the
                                                        ## considered edge
                travelTime = float(listP[0]) + float(listC[2]) + float(wait)
                connected.append([x, travelTime, ID])
    return connected

## This function selects the road segment with the lowest total travel costs from the queue and
## returns that line
```

```
def MinQueue (queue):
    q = queue[0]
    minTT = q[1]
    minLine = q
    for x in queue:
        if x[1] < minTT:
            minTT = x[1]
            minLine = x
    return minLine

## This function returns a list of line ID's that are part of the shortest path by starting with the
## destination line ID and adding the predecessor of that line until the source is reached.

def ReturnRoute(source, dest, dests):
    route = []
    v = dest
    route.append(v)
    while v != source:
        v = dests[v][1]
        route.append(v)
    return route

## This function builds a SQL statement containing all the line ID's that are part of the route.
## The SQL statement is returned

def SelectRoute(path, roads):
    a = int(path[0])
    selstat = '"ID" = ' + str(a)
    path.remove(path[0])
    for x in path:
        a = int(x)
        selstat = selstat + ' OR "ID" = ' + str(a)
    return selstat

## This function calculates new travel costs for the segments that are partly within the plume

def AlterTraveltimes(roads, plume):
    gp.MakeFeatureLayer(roads, "rlayer")
    gp.SelectLayerByLocation_management("rlayer", "INTERSECT", plume, "0 meter",
        "NEW_SELECTION")
    gp.CalculateField_management("rlayer", "MINUTES", "[St_Travel]* 10")

## This function calculates the original travel costs of each segment and stores it in the travel
## costs field

def ResetTraveltimes(roads):
    gp.CalculateField_management(roads, "MINUTES", "[St_Travel]")

# preparations before the start of the algorithm
```

```
import arcgisscripting, string

gp = arcgisscripting.create(9.3)

gp.overwriteoutput = "1"
gp.workspace = r"D:\Dijkstra"
roads = "[roads].shp"
sourcepoint = "Startpoints.shp"
destpoint = "Endpoints.shp"
plume = "[plume].shp"
UnitID = [unitnumber]

## set the travel costs of the roads in the network that are crossed by the restricted area
AlterTraveltimes(roads, plume)

## determine the line ID's of the lines the source and destination location are located on
source = determineLine(UnitID, sourcepoint)
dest = determineLine(UnitID, destpoint)

print "position source and destination determined"
print "collecting road segments"

## create the lists and dictionaries needed during the determination of the route
Graph = createRoadDictionary(roads)
queue = []
dests = {}

print "road segments collected"

## set initial values; the line the source is on gets a travel cost of zero and has no predecessor
ID = source
dests[ID] = [0, 0.0]

# start of algorithm

print "calculating route..."

## for the current edge the connected edges are investigated. The connected edge with the
## lowest total travel costs is set as the current edge for which the connected edges are
## identified. This process continues until the destination edge has become the current edge.

while ID != dest:
    con = findConnectedRoads(ID, Graph, dests, queue)
    for x in con:
        queue.append(x)

    min = MinQueue(queue)
    dests[min[0]] = [min[1], min[2]]
    queue.remove(min)
```

```
    ID = min[0]
    print int(ID)
print dests

## When the lowest travel cost to the destination edge has been found, the path is determined
## and the total travel cost is stored in the estimated time of arrival (ETA) variable

print "collecting results..."
ETA = dests[dest][0]
path = ReturnRoute(source, dest, dests)
print path

## A selection statement is created to make a selection in the map of all the edges that
## participate in the shortest path

selstat = SelectRoute(path, roads)
print selstat

## the travel costs that were altered to incorporate the restricted area are changed back into
## their original value to prepare for the next run of the algorithm.

ResetTraveltimes(roads)
gp.MakeFeatureLayer(roads, "Rlyr", selstat)
gp.CopyFeatures_management("Rlyr", "route.shp")

del gp
print "route determined"
print "The total travel time is " + str(ETA) + " minutes."
```