Modeling and Assessing Crossing Elimination as a Strategy to Reduce Evacuee Travel Time

Arash Jahangiri

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science
In
Civil and Environmental Engineering

December 12th, 2012
Blacksburg, VA

Keywords: Crossing elimination, evacuee travel time, simulated annealing algorithm, DynusT

Modeling and Assessing Crossing Elimination as a Strategy to Reduce Evacuee Travel Time

Arash Jahangiri

ABSTRACT

During evacuations, emergency managers and departments of transportation seek to facilitate the movement of citizens out of impacted or threatened areas. One strategy they may consider is crossing elimination, which prohibits certain movements at intersections, that may be permissible under normal operating conditions. A few previous studies examined this strategy in conjunction with contra-flow operations, but fewer have considered crossing elimination by itself. This study helps fill the existing gap in knowledge of the individual effects of crossing elimination.

A bi-level model that iterates between optimization and simulation is developed to determine the optimal configuration of intersection movements from a set of pre-specified possible configurations for intersections in a given area. At the upper level, evacuees' travel time is minimized and at the lower level, traffic is assigned to the network with the traffic assignment-simulation software DynusT. The overall model is solved with a simulated annealing heuristic and applied to a real case study to assess the impact of crossing elimination.

Three scenarios are developed and examined using the solution method proposed in this research. These scenarios are developed using combinations of two elements: (1) Evacuee destination distributions, and (2) Evacuee departure time distributions. Results showed about 3-5 percent improvement in total evacuee travel time can be achieved in these scenarios. Availability of through movements at intersections and existing merging points in movement configurations are the two factors influencing the selection of movement configurations.

Acknowledgement

I would like to thank my committee members, Dr. Pamela Murray-Tuite, Dr. Kathleen Hancock, and Dr. Hazhir Rahmandad for their help and invaluable suggestions. This thesis would not have been possible without the guidance and the help of this committee. I am especially thankful for Dr. Murray-Tuite's help and support throughout all parts of this work.

I also would like to appreciate my wife for being so supportive throughout my studies. I also would like to express my appreciation to my family for their support and encouragement from hundreds of miles away, and finally many thanks to my friends and colleagues: Nop, Weihao, and Sirui, who all have helped me many times.

Table of contents

List of figures

List of tables

Chapter 1 Introduction

1.1 Background

Hurricanes, tornados, floods, earthquakes, tsunamis, volcanic eruptions, power plant incidents, chemical spills, fires, terrorist attacks, etc. can take hundreds and thousands of lives in a single event, and now more lives are threatened by natural and man-made disasters compared to the past. Increasing numbers of natural and man-made disasters and the possible terrible consequences have led planners to develop regional evacuation plans. Statistics demonstrate that natural and man-made disasters have been increasing as shown in figure 1.

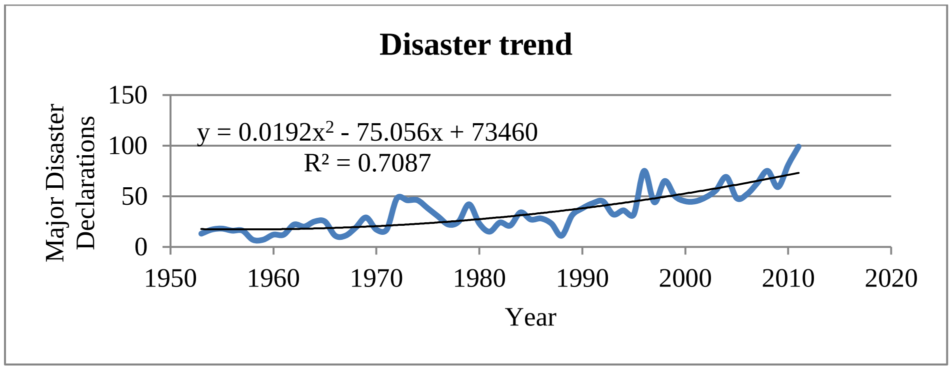


Figure 1 Increasing number of disasters in the US (FEMA: http://www.fema.gov/news/disaster_totals_annual.fema)

Some of these catastrophes such as hurricanes and floods can be anticipated hours or even days before they occur. However, no reliable method has been found to anticipate some other disasters like earthquakes. People have always been seeking to foresee the future to be prepared for crises. The first prediction was officially made in the United States in 1948 about an upcoming tornado. Hurricane predictions became reliable in 1960 using geostationary satellites. Thanks to the technology, great improvements were achieved in recent years in predicting all different kinds of disasters. In cases of predictable disasters, minutes and even seconds of an earlier warning can save many lives (Pielke 1998).

Emergency situations have some specific characteristics. During evacuations, a large amount of people try to leave in a very short time, so congestion and heavy delays are experienced throughout the road network and trip destinations might be different from normal trips. The extreme resulting delays have made engineers, planners, and emergency management agencies consider network management strategies such as ramp metering and closure (McGhee and Grimes 2006; Fonseca, Moynihan et al. 2009; Friso, Van Zuilekom et al. 2009; Edara, Sharma et al. 2010; So and Daganzo 2010; Friso, van Zuilekom et al. 2011), crossing elimination (Cova and Johnson 2003; Kalafatas and Peeta 2009; Xie, Lin et al. 2010; Xie and Turnquist 2011), and contraflow operations (Wolshon 2001; Theodoulou and Wolshon 2004; Theodoulou and Wolshon 2004; Tuydes and Ziliaskopoulos 2004; Patel 2006; Tuydes 2006; Tuydes and Ziliaskopoulos 2006; Wolshon 2006; Chen and Chou 2008; Liu, Chang et al. 2008; Chen and Chou 2009; Kalafatas and Peeta 2009). Ramp closure and crossing elimination strategies can smooth the flow of traffic and can speed up the process of evacuation from threatened areas toward safe zones by preventing traffic from using some ramps in case of ramp closure, and by restricting some movements at certain intersections in case of crossing elimination. The contraflow strategy is intended to maximize the capacity of the road network from threatened areas toward safe zones by changing the direction of traffic lanes.

The focus of this thesis is on developing a model to apply crossing elimination as a strategy to reduce evacuee travel time.

1.2 Motivation

Crossing elimination is a relatively new strategy that can be considered during evacuations and it has been usually evaluated together with contraflow operations. Therefore, the benefits provided by this particular strategy are unclear. When combining two or more strategies (Crossing elimination and contraflow in this case) at the same time, it probably leads to a better outcome. Nevertheless, it is always desirable to understand strategies' benefits individually. Moreover, contraflow takes a long time to set up. In addition, sometimes for smaller evacuation areas contraflow is not needed. Hence, this study concentrates on the crossing elimination strategy by itself to identify the benefits exclusive to this strategy.

The primary goal of this strategy is to minimize or reduce vehicle conflicts at intersections to reduce delay and consequently to decrease evacuation time. During emergency evacuations, heavy traffic delays are experienced at both freeways and primary intersections. The delays at intersections can be caused by two types of conflicts: crossing conflicts that are more important and influential and merging conflicts(Cova and Johnson 2003). To minimize conflicts, selected movements at intersections can be eliminated. Figure 2 shows a simple example of a crossing elimination plan. Twenty potential crossing and merging conflicts are decreased to only eight conflicts when restricting all left turns of all approaches.

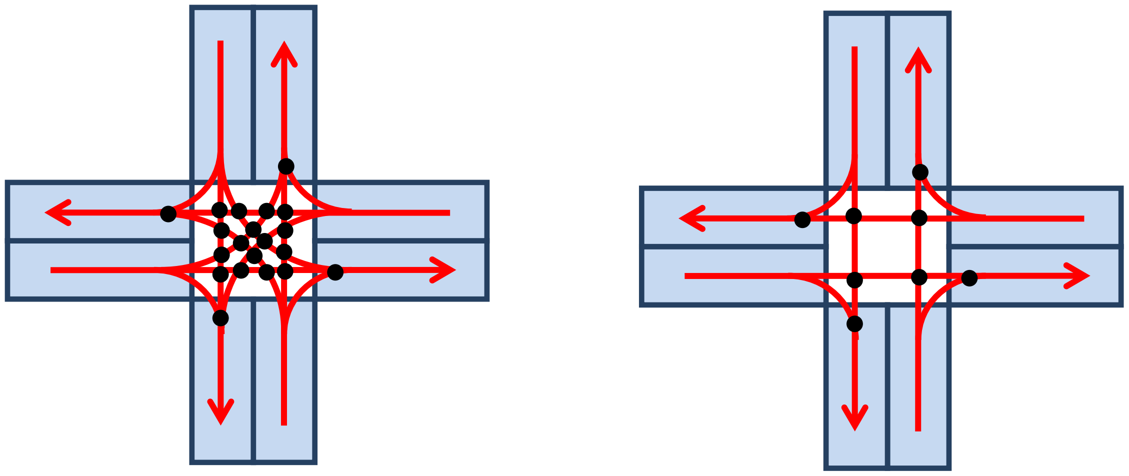


Figure 2 no action vs. restricting left turns

Implementing crossing elimination would lead to three advantages. First, outbound traffic flow would increase at intersections. Second, it channels traffic flow along certain routes and increases traffic safety. A higher level of safety is achieved since the conflicts are decreased and lane changing behavior is decreased. Third, it would be beneficial when the traffic signal and communication system fails which is likely to happen when disasters occur (Xie, Lin et al. 2010).

1.3 Research complexity

In relatively large networks, it is a complicated decision making process to determine movement restrictions since it is not easy to identify which intersections would be favorable and which movements of those intersections should be restricted. After applying movement restrictions at an intersection, one may not be able to turn left or right or even go straight at that intersection depending on the type of restriction. Therefore, this strategy may make some people drive longer distances, but can reduce total evacuee travel time since they can drive more smoothly with higher speeds. The reason why people can move more freely and quickly is that the traffic flow becomes uninterrupted at the intersections at which crossing elimination is applied. However, it may take some people more time to evacuate. It might happen that, even though one can drive with higher speed because of the smoother traffic flow, the distance traveled is increased to the point that even the higher speed cannot compensate (Cova and Johnson 2003). Nevertheless, in the optimization procedure of these kinds of problems, although travel time of some evacuees may increase, the strategy is adopted in a way that the total travel time of network evacuees decreases. Thus, determining the target intersections and desired movement configurations that would contribute to the overall improvements creates a complex problem, which is a combinatorial optimization problem that requires high computational effort to solve.

1.4 Significance of the research

Only a few previous studies examined this strategy, and most of those studies have considered this strategy combined with contra-flow operations. The present study helps to better understand the advantages of crossing elimination strategy by itself.

Furthermore, previous literature was unclear about considering evacuee travel time which is more important than the non-evacuee travel time. Only a few studies have separated evacuees from background traffic and considered the evacuee travel time as their objective function (e.g. (Chiu and Mirchandani 2008)). In most evacuation studies the following measures are evaluated as the objective function: Total evacuation time as in (Xie and Turnquist 2009; Ng, Park et al. 2010; Xie, Lin et al. 2010; Xie and Turnquist 2011), total travel distance as in (Cova and Johnson 2003), total travel time of all travelers as in (Han, Yuan et al. 2006; Kalafatas and Peeta 2009). Even though minimizing these measures might be a decent objective and is usually used throughout the literature, it is more important to obtain evacuee travel time as part of total users. Therefore, in this research, evacuation traffic is distinguished from the background traffic and the travel time of total evacuees is minimized as the objective function.

1.5 Research objectives

This research aims to:

- Develop an optimization heuristic approach to assess crossing elimination as a strategy to reduce evacuee travel time without using any other strategies to understand the benefits of this strategy by itself.

- Separate evacuation traffic from background traffic and consider only evacuee travel time in the objective function.
- Simulate the strategy on a large network as a case study for no-notice evacuation scenarios.

1.6 Thesis outline

Five additional chapters are presented in the thesis. Pertinent research conducted in the area of emergency evacuation strategies, specifically crossing elimination, are reviewed in the second chapter. In the third chapter, the problem is modeled and formulated as a bi-level model. Next, a simulated annealing (SA) algorithm as a heuristic technique is discussed and presented as the solution approach. Chapter five applies the crossing elimination strategy to a road network as a case study and presents the results. Finally, conclusions as well as future research are provided in the last chapter.

Chapter 2 Literature review

2.1 Background

Increasing number of catastrophes and their terrible consequences have contributed to the consideration of a variety of network management strategies, including ramp metering and closure, contraflow operations, and signal timing optimization. These strategies are briefly described in this section.

2.1.1 Ramp metering and closure

Ramp metering as studied by (McGhee and Grimes 2006; Fonseca, Moynihan et al. 2009; Friso, Van Zuilekom et al. 2009; Edara, Sharma et al. 2010; So and Daganzo 2010; Friso, van Zuilekom et al. 2011) can be adopted to control the traffic flow on freeways by deciding whether or not to fully or partially close ramp entrances. Ramp closure can be considered an extreme of ramp metering where the entrance ramps are fully closed and the traffic flow rate is zero from those ramps. The problem is to decide which ramps to close that would bring the most benefit. The decision making process can be developed as an optimization problem where the decision variables are integer variables that indicate whether the ramp is open or closed, as in Fonseca et al (2009). The purpose of ramp metering/closure is to smooth and speed traffic flow through the network from danger zones towards safe areas and as it can be inferred, these strategies are typically used on freeways (Ghanipoor-Machiani, Murray-Tuite et al. 2013).

2.1.2 Contraflow operation

Many studies (Wolshon 2001; Theodoulou and Wolshon 2004; Theodoulou and Wolshon 2004; Tuydes and Ziliaskopoulos 2004; Patel 2006; Tuydes 2006; Tuydes and Ziliaskopoulos 2006; Wolshon 2006; Chen and Chou 2008; Liu, Chang et al. 2008; Chen and Chou 2009; Kalafatas and Peeta 2009) have considered contraflow operation to increase the capacity of road networks in favor of evacuation directions. Contraflow operations may be considered for any road type, but in most cases freeways are applied. Their purpose is to increase the number of lanes in the evacuation direction. In other words, a huge amount of demand exists during evacuation and there is a dominant direction people need to take which is from the danger zones toward the safe areas. Therefore, as the road network is not designed for this purpose it lacks sufficient capacity in the evacuation direction. Contraflow strategy tries to change the direction of roads in the favor of increasing the capacity of the evacuation direction. Contraflow lanes have been considered together with crossing elimination strategies, as in Kalafatas and Peeta (2009) and (Xie, Waller et al. 2011).

2.1.3 Signal timing optimization

Research studies have been conducted to apply this strategy to minimize evacuation time (Miller-Hooks and Tarnoff 2005; Ming, Lichun et al. 2007; Liu, Chang et al. 2008; Ren, Huang et al. 2012). When disasters occur, trip origins, destinations, as well as traffic volumes differ from the normal conditions. Therefore, to reflect the changes, current signal timing plans, especially fixed plans, need to be updated in accordance with the new situations. If the way intersections are controlled is not updated, intersections can negatively impact the evacuation as they have the potential of becoming serious bottlenecks. Widespread interviews with experts and agents in federal, state and local agencies were carried out by Miller-Hooks and Tarnoff (2005) in the United States between September 2004 and February 2005. These interviews illustrated that there were four approaches to modifying signal timing: (1) set signals on flash (2) control signals by police at critical intersections (3) set signals on PM-peak setting (4) set signal timing plans on maximum cycle length on evacuation routes, giving the majority of green time to the major directions.

Crossing elimination is combined with identifying the locations where traffic signals need to be updated. A bi-level model is presented to identify the locations where signal timing needs to be modified and the locations where crossing elimination needs to be implemented. It should be pointed out that only the optimal locations are identified where signals are needed, however, the signal timings are not optimized. Identifying these locations as well as the movement restrictions at those uninterrupted flow intersections can maximize the operational efficiency during evacuations (Liu and Luo 2012). Crossing elimination and signal timing optimization are similar as they both try to minimize the delays at intersections. However, crossing elimination has the advantage as mentioned earlier when traffic signal and communication system fails which is likely to happen when disasters occur.

2.2 Crossing elimination

Crossing elimination concentrates on how movements are controlled at intersections and tries to identify the best movement configurations for intersections that bring the most benefit which is to minimize evacuees' travel time. Past studies regarding crossing elimination which is a relatively newer strategy and the focus of this thesis, are reviewed in this section.

One of the first studies that considered the crossing elimination strategy was conducted by Cova and Johnson (2003) who called it a lane-based routing strategy. To identify the optimal lane-based evacuation routing plan a network flow model was presented. The model aims at reducing delays at intersections by restricting some turning movements with the purpose of directing the traffic flow toward safe areas. Smoother vehicle movements, capacity growth for the desired direction, and conflict reduction can be achieved because the traffic flow is converted from interrupted to uninterrupted flow. A linear program was developed to present the problem as a minimum-cost flow problem. The network simplex technique is a method that can be used to solve this problem. The evacuation routing problem is actually an integer extension of the minimum-cost flow problem, which minimizes total travel distance with regard to a couple of restrictions: flow conservation constraints, prevention of intersection crossing conflicts, restricting the total number of merges allowed, and capacity limit (Cova and Johnson 2003).

Crossing elimination is combined with contraflow operation by Xie and Turnquist (2009). They proposed a bi-level optimization model from the combination of eliminating intersection crossings with contraflow lanes and reserving lanes for emergency vehicles. The idea was to create uninterrupted traffic flows to help people move more quickly. They converted every intersection as well as roadway sections into sub-networks to model the evacuation network design problem. The upper level problem minimizes total network evacuation time and the lower level problem assigns traffic on the road network, which essentially was a probit-based stochastic traffic assignment problem. Lagrangian relaxation and tabu search were used to simplify and solve the problem. It was assumed that evacuees choose their routes and destinations simultaneously. Constraints included reserving a sufficient number of lanes for emergency vehicles.

The combination of lane reversal and crossing elimination was pursued and extended in a similar approach by Xie, Lin, et al. (2010) in a dynamic evacuation network optimization problem. Bi-level modeling was again conducted. Minimizing total evacuation time and network clearance time were the two objectives that were considered in the upper level problem, and the constraints were similar to their 2009 work. The lower level problem is a single destination user optimal dynamic traffic assignment (DTA) model. They solved the problem with an integrated Lagrangian relaxation and Tabu search method (Xie, Lin et al. 2010). These solution techniques were also used in Xie and Turnquist's (2011) work.

Bretschneider and Kimms (2011) developed a dynamic flow model for their evacuation problem. They also used relaxation based heuristics in their solution approach. Link-based vehicle flows and the number of lanes used were considered as the decision variables. For the purpose of capturing variations among different time intervals, departure and arrival times of the vehicles were taken into account. As the objective function, the average evacuation time was used. In addition to flow enforcement, flow conservation, and capacity constraints, several constraints were used for lane consistency, different crossing conflicts prohibition, and avoidance of irregular flow behavior.

Xie, Waller et al (2011) combined crossing elimination and contraflow strategies once more with a different solution approach. They presented a mixed linear integer programming model and similar to the 2009 and 2010 works, they converted intersections into a sub-network in which all movements are represented by different links. The goal was to find an intersection origin-destination flow pattern that minimizes the number of traffic crossing points while considering inbound and outbound traffic flows in the newly defined sub-network. They indicated that the solution technique they adopted, a simplex-based approach, was very efficient.

In the present study, bi-level modeling is used as in several past studies mentioned above, focusing only on the crossing elimination strategy instead of combining it with contraflow strategy, thus identifying the benefits exclusive to this strategy. The solution approach uses a simulated annealing (SA) algorithm which is a heuristic technique and is unique to this study. Table 1 presents key factors regarding past studies as well as the present study.

Table 1 key features of different crossing elimination studies

Study	**Objective function**	**Solution approaches**	**modeling**	**Combined or standalone**
(Cova and Johnson 2003)	Minimization of total travel distance	simplex methods	Minimum-cost flow model	Combined with contraflow
(Xie and Turnquist 2009; Xie, Lin et al. 2010; Xie and Turnquist 2011)	Minimization of total network evacuation time and network clearance time	Lagrangian relaxation and tabu search	Bi-level model	Combined with contraflow
(Xie, Waller et al. 2011)	Minimization of number of crossing points	Simplex-based	mixed linear integer programming model	Combined with contraflow
(Bretschneider and Kimms 2011)	average evacuation time	Relaxation-based heuristic	dynamic flow model	Combined with contraflow
(Liu and Luo 2012)	Minimization of total evacuation time	genetic algorithm (GA) -based heuristic	Bi-level model	Combined with signal optimization[1]
Present study	Minimization of total evacuee travel time	Simulated Annealing and simulation	Bi-level model	Standalone

[1] For the signal optimization part, only optimal locations are identified, the timings are not optimized

Chapter 3 Problem formulation

Before formulating the problem, different classes of areas need to be defined including impacted zones, shadow evacuation zones, and a subarea used in the simulation runs. Furthermore, some modeling considerations such as selection of the objective function and movement configurations need to be explained.

3.1 Study areas

As shown in figure 3, the green dot represents the center of danger. The study area under consideration can be defined as follows: the impacted zones (G''') that are in immediate danger and are required to be evacuated, shadow evacuation zones (G'') that are farther from the point of danger, but still some people from these zones evacuate although it is not needed, and a subarea (G’) from the entire network (G) that includes G’’ plus an extension to capture the realistic decision making of road users in response to the restrictions coming from the crossing elimination strategy. As illustrated in figure 3, the rest of the road network beyond G’ is considered to be unaffected and is removed completely from the study area to lessen the computations.

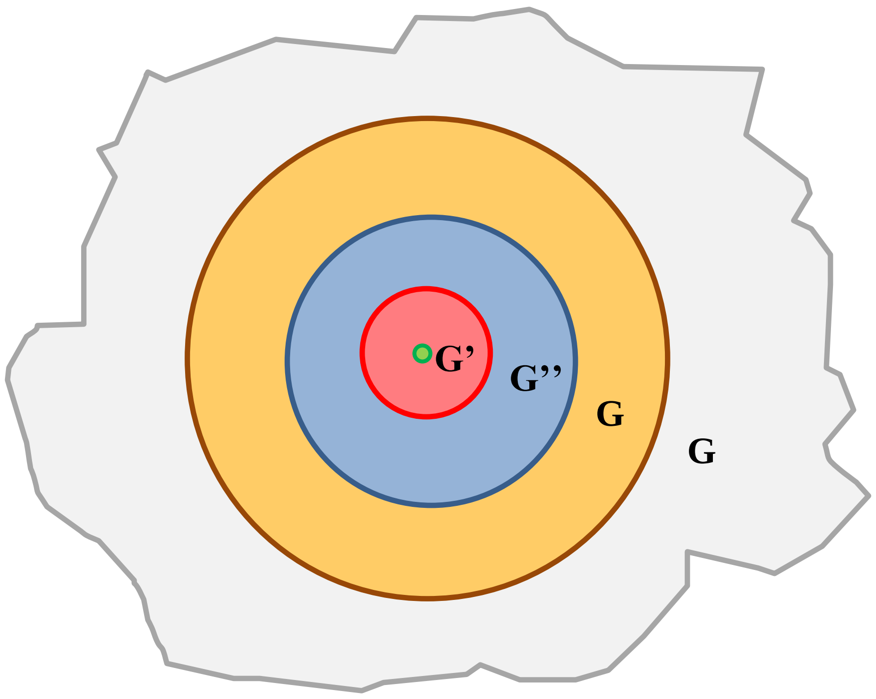


Figure 3 study areas under consideration

The rationale behind the extension of study area from G’’ to G’ is explained as follows. After changing movement configurations at intersections, a driver might take a new route that may not exist in the G’’ area. Figure 4 shows an example of how a movement configuration can affect drivers’ decisions on taking routes and why the extension of the study area from G’’ to G’ needs to be included. A vehicle (the brown rectangle) is on road 2 approaching an intersection. The driver’s initial decision is to go through the intersection remaining on road 2. However, assuming that the depicted movement configuration is selected for that intersection, the driver must take a right turn at the intersection and continue driving on road 1. Without G’ included in the study area, motorists in the simulation model like the vehicle in the example are not able to continue their trips as they get stuck on the G’’ boundary.

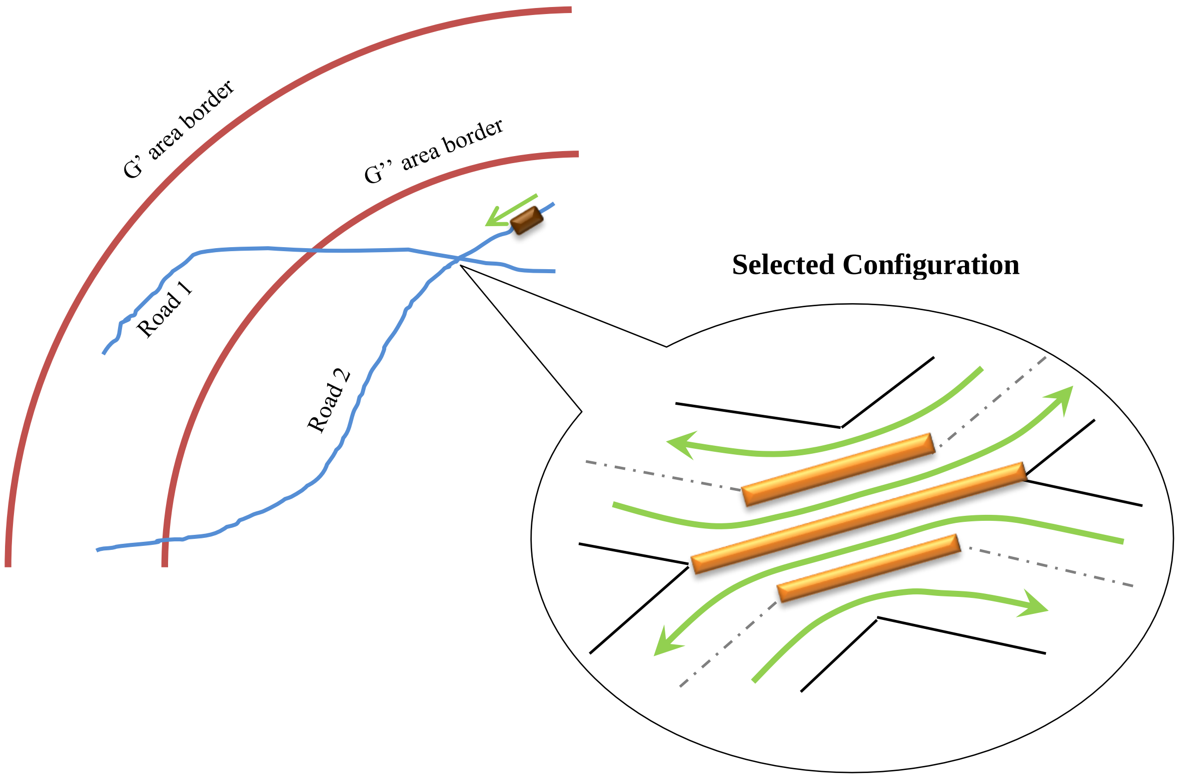


Figure 4 study area extension – reasoning

Based on this description, the problem of interest can be formally stated as: given a transportation network G(N, A) consisting of a set of nodes N and directed arcs A, with a mandatory evacuation, G’’’ area, and anticipated shadow evacuation, (G") area, and a subarea of this network G' (N', A'), find the set of movement restrictions M at intersections J ($J \subset N''$) within the area G'' (N'', A'') such that the total travel time for the evacuees in the mandatory evacuation area G''' (N''', A''') is minimized.

3.2 Modeling considerations

The problem can be formulated as a bi-level model; the objective function of the upper level problem is to minimize evacuees' total travel time. To obtain the objective function, G''' is examined because zones within this area are defined as impacted zones. Thus, the objective function of the upper level problem is to minimize total travel time of people whose trips are originated from G''' area and destined to outside this area (anywhere outside G'''). However, the past literature is unclear about the consideration of only evacuee travel time in the objective function which is more desirable than including the travel time of all road network users. The lower level problem assigns traffic to the road network using DynusT which is simulation-based dynamic traffic assignment (DTA) software through which the travel times are determined.

It appears that the reason why crossing elimination has usually been combined with the contraflow operation is that they are generally related to each other, meaning that restricting movements at an intersection can impact the direction of traffic flow on the approaches of that intersection, which is what the focus of contraflow operation is. Similarly, direction changes in traffic flow influences the way movement configurations can be restricted at intersections. Hence, as this study only evaluates crossing elimination strategy, possible movement restrictions that change the direction of traffic at the intersection approaches are not taken into account. Only those movement configurations that preserve the normal two way direction of all approaches need to be assessed.

3.3 Model formulation

The problem can be formulated as below.

Upper level problem:

$$\min z = \sum_{d \in D} \sum_{s \in S} \sum_{v \in V} t_v^{ds} \tag{1}$$

s.t.

$$\sum_{i \in I} \sum_{j \in J} m_{ij} c \le B \tag{2}$$

$$\sum_{i \in I} m_{ij} \le 1 \qquad \forall j \in J \tag{3}$$

$$m_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J \tag{4}$$

Lower level: Traffic Assignment Problem using DynusT based on *G' (N', A'), M*

where:

D	is the set of impacted zones, $d \in D$
S	is the set of zones (safe) outside the impacted zones, $s \in S$
V	is the set of vehicles corresponding to the mandatory evacuees, $v \in V$
t_v^{ds}	is the travel time of mandatory evacuating vehicle *v* traveling from zone *d* to zone *s* (determined by the traffic assignment output)
J	is the set of all intersections in N'', $j \in J$
I	is the set of predefined configurations for movement restrictions for set J, $i \in I$
m_{ij}	represent the binary integer decision variables that take the value 1 if configuration *i* is selected for intersection *j* and 0 otherwise
M	is the matrix of movement restrictions
c	is the cost of modifying an intersection
B	is the total budget available for modifying the intersections.

Budget constraint, equation (2), includes the amount of available equipment and number of personnel needed for applying the movement restrictions. Constraint (3) explains that each intersection can be assigned to at most one movement restriction. In other words, one particular intersection cannot have two different configurations at the same time. Equation (4) defines a binary variable that indicates which configuration is assigned to which intersection.

Chapter 4 Solution approach

4.1 Introduction

Dealing with combinatorial optimization problems in which a great number of solutions are possible, may become a very time consuming effort as the huge number of feasible solutions need to be tested to find the optimal one (Michiels, Aarts et al. 2007). The objective functions of these kinds of problems represent a measure showing how efficiently a complex system, which consists of many different parts, can work. Each configuration of those parts actually forms a feasible solution with respect to some restrictions. Thus, these objective functions or cost functions depend on the detailed configuration of different parts of the system (Kirkpatrick, Gelatt et al. 1983).

Simulated annealing (SA) is one of the meta-heuristics that can be adopted to solve such problems efficiently. This technique can be applied to both discrete and continuous optimization problems. However, it has been used mostly for solving discrete problems. The known characteristic of the algorithm is that it does not get trapped in local optima, so while solving the problems, if properly formulated, SA may allow worse solutions to be accepted in the hope that the near optimal solution will be obtained after some future iterations (Glover and Kochenberger 2003).

The model in chapter 3 is solved using simulated annealing as well as DynusT (simulation-based dynamic traffic assignment software). The entire solution approach is based on SA algorithm and DynusT is integrated into a part of SA procedure. The SA algorithm and DynusT are explained in more detail below.

4.2 Simulated annealing algorithm

The Algorithm's steps are summarized as follows (Glover and Kochenberger 2003):

The term "temperature" used to describe this algorithm is basically a control parameter which is used in making decisions on accepting or rejecting new solutions.

Step1. Define an initial solution

Step2. Choose algorithm's setting

- Initial temperature ($t_{initial}$)
- Final temperature (t_{final}) and stopping criteria
- Number of iterations at each temperature (M_k)
- Cooling schedule

Step3. Repeat until stopping criteria are met

- Repeat until n=M_k, n is a counter, starting from 0
    - Generate a new solution
    - Calculate Δ, the difference between new and current objective functions
    - If $\Delta \leq 0$, the new solution is accepted, otherwise, the new solution can be accepted with the probability of $e^{-\left(\Delta/_{temperature}\right)}$
    - n=n+1
- Decrease the temperature according to the cooling schedule

The initial solution could be any feasible solution. Unlike some iterative improvement methods, finding a good initial solution would not be helpful in the case of SA as the initial solution does not impact the final solution. It is recommended not to spend much time on constructing the initial solution (Michiels, Aarts et al. 2007). The current configuration of the system of interest can be a good initial solution as it is already constructed. Also, starting from the current conditions means that the approach starts from a point where no strategy is applied and goes from there and starts making changes (i.e. applying strategies) to see how the starting point can be improved, which is like monitoring how the current situation is improving.

4.2.1 Annealing Schedule

Step 2 of the algorithm which defines initial temperature, final temperature, number of iterations at each temperature, and cooling schedule is called the annealing schedule that needs to be determined in order for SA to run. Generally, two approaches exist in this regard; static and dynamic approaches. For ease of implementation, the static approach in which fixed values of these parameters were chosen before running the algorithm was used (Romeo F 1991).

Generally, the initial temperature should be big enough so the algorithm can search all the solution space. Moreover, if a very low value is chosen for initial temperature, SA may get trapped since it cannot search the solution space very well. On the other hand, a value that is too high leads to wasting some iterations as it accepts almost all worse solutions in the beginning of the algorithm (Glover and Kochenberger 2003). Lian-bo and Feng (2009) suggested a method to get the initial temperature. An initial temperature is chosen first and the algorithm is repeated for n times with the selected temperature. Second, the probability of accepting the new solution is calculated. If the probability is around 0.8~0.9, the selected temperature is set as the initial temperature. If the probability is too high or too low the temperature should be adjusted appropriately, and the steps should be repeated until the required probability for accepting the new solution is obtained.

The goal of the stopping criterion is to identify the stage in which an acceptable solution has been reached, and to recognize that the temperature has become very small so the probability of accepting worse solutions is very low. After reaching a pre-specified final temperature that is set to a small value near zero, a typical criterion to terminate the procedure is applied which is terminating the algorithm once the objective function does not change significantly for a few consecutive temperatures or iterations. Furthermore, to decrease the temperature, the geometric temperature reduction method, in which a coefficient with the value of less than one is multiplied by the current temperature, is a typical technique. The coefficient value is desired to be close to one to have a gradual temperature reduction rather than an abrupt reduction. Moreover, a couple of iterations are carried out at each temperature with respect to the size of the problem (Romeo F 1991).

Considering a particular combinatorial optimization problem, objective functions different in magnitude are dealt with when changing some variables related to that problem. For example, in transportation problems, in case we choose total travel time of network users as the objective function, when changing some variables such as time of day, demand, road capacities, etc the objective function also changes. As a result, the difference between objective functions of different conditions may change. In other words, considering the term $e^{-\left(\Delta/_{temperature}\right)}$, the magnitude of changes in objective function, denoted by Δ, can affect the acceptance probability of worse solutions so much and make it difficult to determine the initial temperature. In fact, choosing a suitable initial temperature depends on those changes. In order to avoid this issue, the relative difference between the new and current objective function is considered in this study as follows.

$$\Delta C_{ij} = (C_j - C_i)/C_j \qquad (5)$$

where,

ΔC_{ij}:	Relative difference of objective function values
C_j:	Objective function of the new solution *j*
C_i:	Objective function of the current solution *i*

4.3 DynusT

Dynamic Urban Systems for Transportation (DynusT) is simulation-based dynamic traffic assignment (DTA) software that was developed recently mainly by researchers at the DynusT Laboratory at the Department of Civil Engineering and Engineering Mechanics (CEEM) at the University of Arizona that can be used for transportation planning and traffic operations to deal with transportation problems.

Using DynusT, traffic volumes and travel times are calculated through DTA models and are used to obtain the objective function value of the upper-level problem. DynusT offers the following features:

- It can capture the complex and dynamic interactions between supply (Transportation modes, network configurations, traffic controls) and demand (participation, departure times, destination and routes).
- It applies DTA models to reflect realistic behavior of users. DTA models account for variations in time, meaning that travel time on a link may vary as time passes
- A built-in tool can be used to cut a subarea of interest from the entire road network
- It has the capability of including traveler information systems such as variable message signs (VMS)

4.4 Solution steps

As depicted on the flowchart in figure 5, the SA algorithm iterates between the upper and lower level problems. To provide the connection between upper and lower level problem which is basically a connection between the optimization and the simulation, a code is written in Matlab that calls the simulation tool at each iteration to assign traffic and obtain the objective function. Also, the whole solution procedure is coded in Matlab environment. The entire Matlab code can be found in appendix A.

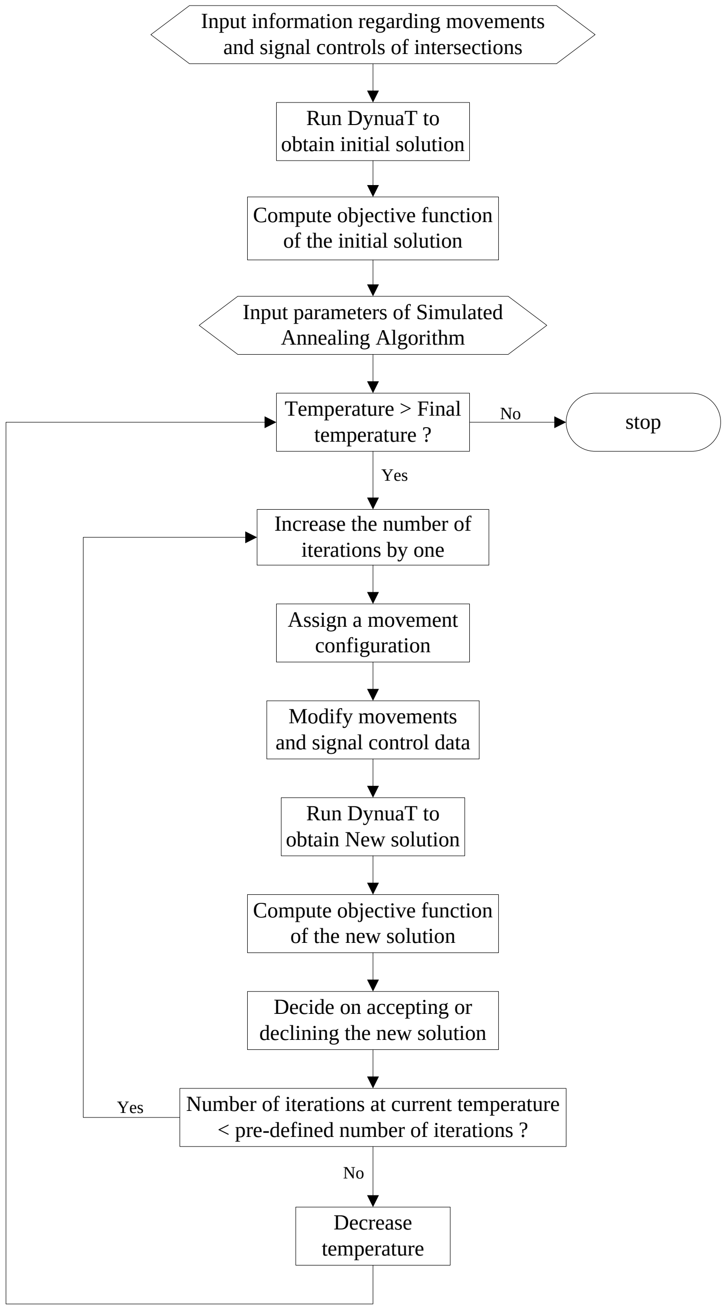

Input information regarding movements and signal controls of intersections
Run DynuaT to obtain initial solution
Compute objective function of the initial solution
Input parameters of Simulated Annealing Algorithm
Temperature > Final temperature ?
No
stop
Yes
Increase the number of iterations by one
Assign a movement configuration
Modify movements and signal control data
Run DynuaT to obtain New solution
Compute objective function of the new solution
Decide on accepting or declining the new solution
Number of iterations at current temperature < pre-defined number of iterations ?
Yes
No
Decrease temperature

Figure 5 Flowchart of solution steps

Chapter 5 Applied Results

5.1 Case study

5.1.1 Network subarea

High computations are associated with running the simulation software (DynusT), specifically when running big networks. To lessen the computational costs, and as the focus is on the above mentioned study areas, the entire road network beyond G’ area was cut to have a much smaller network and as a result the simulation time can considerably decrease. A DynusT functionality called subarea cut was used to remove this area. This tool produces new demand files for the new subarea according to simulation results of the entire network with the original demand files. In other words, the simulation for the entire network needs to be executed first. Then, the subarea tool is able to cut the network once the subarea boundary is defined (which is the G’ area), and after that, it generates new demand files for the G’ area.

5.1.2 Study area assumptions

The crossing elimination strategy is tested on the Northern Virginia road network with the Pentagon as the center of danger. Figure 6 shows the entire road network. Based on the Metropolitan Washington Council of Governments' (MWCOG) transportation planning model, table 2 presents number of nodes, links, and traffic analysis zones (TAZs) associated with the entire network as well as the subarea used for simulation runs. Different study area boundaries are determined by policy makers as follows. Impacted zones are considered as those zones inside a 2-mile radius from the Pentagon that are located within the G''' area. Shadow evacuation zones are assumed to be between a 2-mile radius and a 5-mile radius (G’’). According to Guterbock et al. (2009), 40% of workers and 20% of residents (within G’’)are considered as shadow evacuees and the rest just follow their normal trips. Moreover, an additional 3 mile buffer within the distance of 5 and 8 miles around the Pentagon is added to the 5-mile radius to form the G’ area. The remaining part of the road network is removed using the DynusT subarea tool. Hence, the 5-mile radius is considered as the area that movement configurations are being changed, but the simulation is executed for the 8 mile radius.

Table 2 Characteristics of the networks

Network	Number of links	Number of nodes	Number of TAZs	Number of vehicles loaded (million)
Entire	12890	5648	1240	~5.5
Subarea	3325	1523	372	~1.5

To summarize, three study areas are defined as shown in figures 6 and 7: (a) G''' area; that includes impacted zones that were defined as zones inside the 2-mile radius and within the red boundary as shown in figure 7 with mandatory evacuation order and are used to obtain the objective function, (b) G'' area; shadow evacuation zones that are between the 2-mile and 5-mile radius and within the blue boundary as shown in figure 7. Intersections within this area are candidates for changing the movement configurations, (c) G' area; this area is a subarea of the whole network and is used for simulation runs and includes zones 8 miles from the Pentagon and within the brown boundary as shown in figure 7.

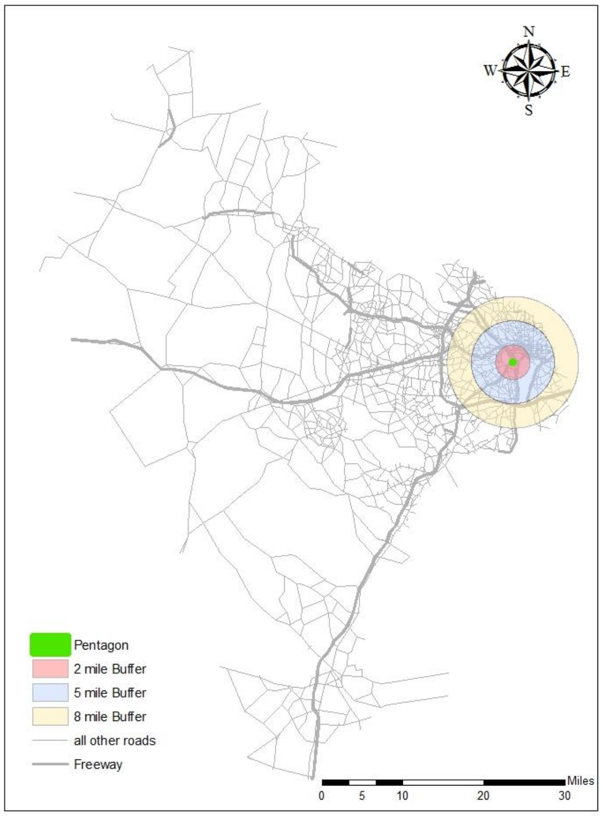


Figure 6. **Study areas around Pentagon**

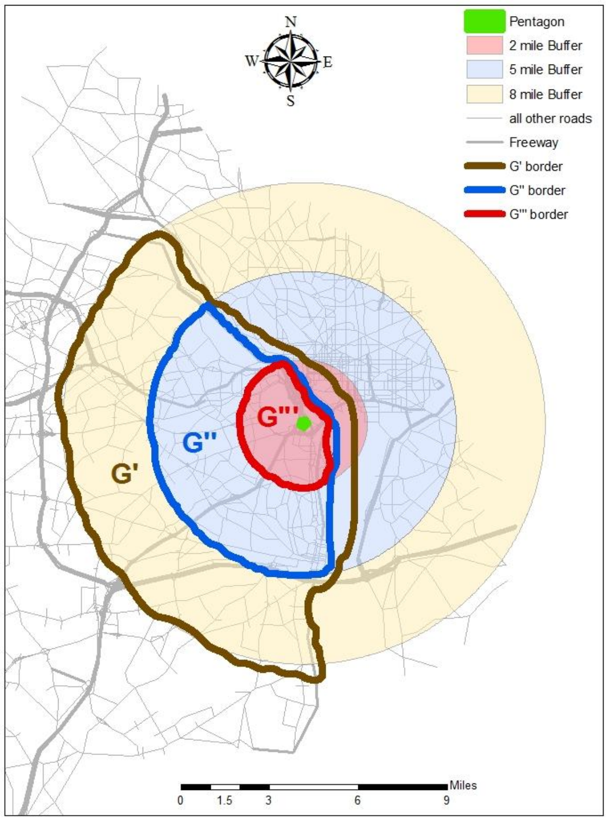


Figure 7. G''': area under consideration for obtaining objective function, G'': area within which intersections are considered for configuration changes, G': area for simulation run

5.1.3 Events timeline

The following table shows the time when different events occur in the order.

Table 3 Events timeline

No.	Time of day	Events
1	3:00 PM	Simulation starts
2	3:45 PM	disaster occurs
3	4:00 PM	Evacuation demand starts
4	4:10 PM	HOV lanes open to all traffic
5	4:10 PM	Crossing elimination strategy is applied
6	12:00 AM	Simulation ends

It is assumed that the disaster occurs at 3:45 PM and the evacuation demand starts 15 minutes later at 4:00 PM. The simulation starts at 3:00 PM, 45 minutes earlier than when the disaster occurs, to warm up the road network as the network is empty at the very first point of the simulation. It is desired to open HOV lanes to all traffic during evacuation to utilize the maximum capacity. It is assumed that they can be opened at 4:10 PM. It is also assumed that it takes 25 minutes to implement movement configurations at the identified intersections, so the crossing elimination strategy is in place at 4:10 PM. Finally, the simulation ends at 12:00 AM.

5.1.4 Movement configurations

Among all types of intersections (three-legged, four-legged, five-legged and above), only four-legged intersections are considered as the candidate intersections since they are the most common type of intersections. Movement restrictions at the selected intersections are chosen from a set of six pre-specified movement configurations as shown in figure 8.

5.1.5 Other assumptions

- Only personal vehicle trips made are considered in this study. Evacuation demand is based on estimates of the number of workers, residents, and tourists/shoppers, developed based on MWCOG's evening peak period origin-destination (OD) matrix and unemployment and retirement rates/age from the 2010 US Census.
- Background traffic includes trips other than evacuation demand, which include non-shadow evacuees within G'' and all trips originated within G'. Background trips follow their habitual paths assuming that the conditions are like a typical day until any movement restriction defined by crossing elimination strategy is encountered.

- At every 10 minutes the drivers are provided with real-time traffic information through radio so they can consider changing their current paths. It is assumed that 100% of evacuees and 0% of background traffic consider the real-time information.
- As far as the budget is concerned, it is assumed that unlimited resources are available to implement different configurations at the candidate intersections. So, the budget constraint in the case study does not have a role in the formulation.

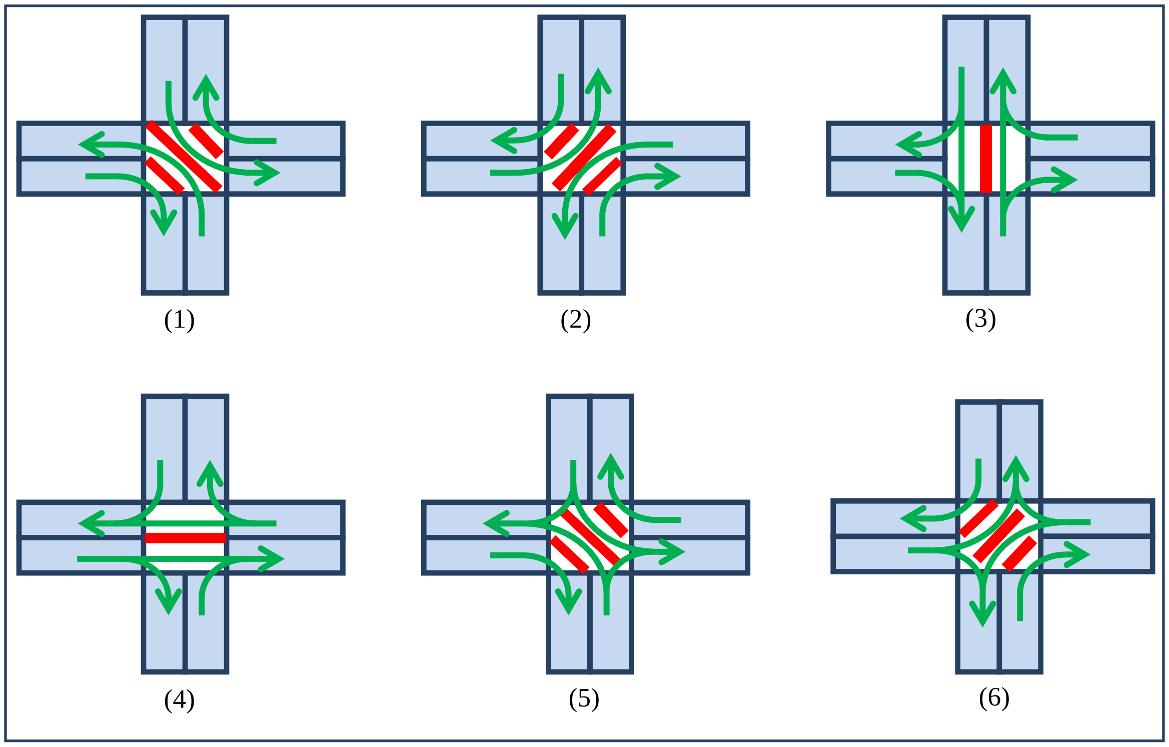


Figure 8 predefined movement configurations at intersections

5.2 Scenarios description

Three scenarios, namely D1T1, D1T2, and D2T1 are examined in this study. These scenarios are a subset of scenarios developed by Murray-Tuite, Park et al. (2012). The following two elements are used to develop the scenarios. In all of these scenarios the departure time of background traffic follows the same pattern as a typical day.

5.2.1 Evacuee destination distributions (D1 and D2)

In the case of D1, trips are distributed throughout the network following MWCOG’s planning model proportions. In this case, people stay with their friends/relatives and/or hotels/motels. In the case of D2, people choose to go to the shelters as their new destinations

5.2.2 Evacuee departure time distributions (T1 and T2)

Several variables have been identified to have impact on the pace of evacuation demand loading. In this study the well-known sigmoid function (Radwan, Hobeika et al. 1985) as shown in equation (6) is used to create two different distributions. In both cases, α is 0.5. The half loading time is assumed to be 30 and 60 minutes for T1 and T2 respectively.

$$P(t) = 1/1 + e^{-\propto(t-T)} \tag{6}$$

Where,

$P(t)$: Cumulative percentage of total volume loaded on the network at time *t*

$\propto$: S-curve slope factor

T: Loading time factor

5.3 Implementing the solution approach

5.3.1 SA algorithm implementation

The program starts with a feasible solution which is sometimes selected randomly or the existing condition can be chosen as well. In this study the existing condition is considered as the starting point and the objective function is monitored and compared with the objective function associated with the starting point to see how effectively the program works.

As mentioned earlier, the annealing schedule needs to be defined to implement the SA algorithm. Initial temperature is selected at first based on the method used by Lian-bo D (2009). However, while monitoring the objective function, the objective function worsens at each iteration. Therefore, trial and error method is used to see at what initial temperature improvements in the objective function can be obtained. After several trials, 0.1 is determined as the initial temperature. Using trial and error again, the final temperature and the number of iterations at each temperature are determined to be 0.01 and 50 respectively. The final temperature is small enough to avoid accepting worse solutions. For the stopping criteria, other than the final temperature consideration, if the improvement in the objective function is less than 0.01 and the acceptance of worse solutions is less than 5 percent for two consecutive temperature stages, the solution algorithm stops. The typical geometric temperature reduction method is adopted to reduce the temperature, and the coefficient of reduction is assumed to be 0.9.

5.3.2 DynusT implementation

There is no specific tool in DynusT to implement crossing elimination strategy. Therefore, the signal timing is adjusted to reflect movement restrictions. No signal timing is needed when using any of the movement configurations. So, in order to implement this for an intersection with a typical two phase signal timing setting, both phases need to be exactly the same allowing specific movements (associated with the selected configuration). Hence, when the second phase starts there is no change compared to the first phase and it simulates the condition as if there is no signal setting with specific movement restrictions. Moreover, according to the events timeline, the strategy needs to be implemented at a specific time, meaning that the entire simulation needs two timing plans; one before the strategy implementation which is the current timing plan and the other after the strategy implementation which is the adjusted plan. All the required information regarding signal settings, number of plans and different movements at intersections are stored in a text file named control.dat. The Matlab code written in this study is able to read this file and modify it accordingly as described.

When motorists encounter a movement restriction at an intersection, they get stuck if their assigned travel path includes that movement which is now restricted. To cope with this issue Variable Message Signs (VMS) are placed on all approaches of the intersection at which a movement configuration is applied. With specific VMS settings, vehicles are rerouted at the intersection of interest. Thus, no one would get stuck. A Matlab function is written to implement VMS as needed.

5.3.3 Computer type

A single simulation run for all scenarios takes around 45 minutes using a server of the following specifications:

- Dell PowerEdge R715
- Two AMD Opteron 2.5 GHz 12 core CPUs having a total of 24 cores
- 256 GB of RAM

5.4 Results and discussion

After running scenarios, total evacuee travel time, selected intersections, and selected movement configurations are evaluated. The number of times each intersection is selected and number of times each movement configuration is selected in all scenarios are considered to identify critical intersections and configurations. Also, other than the total evacuee travel time, average travel times and number of vehicles that fail to complete their trips within the simulation period are assessed as other measures to compare different scenarios.

5.4.1 D1T1 scenario

As illustrated in figure 9, 4.5% improvement is gained with this scenario. In other words, the total evacuees' travel time (14751555.77 minutes) decreases by 4.5 %. A total of 11 intersections with the associated configurations are identified through the solution approach to apply crossing elimination as shown in table 4.

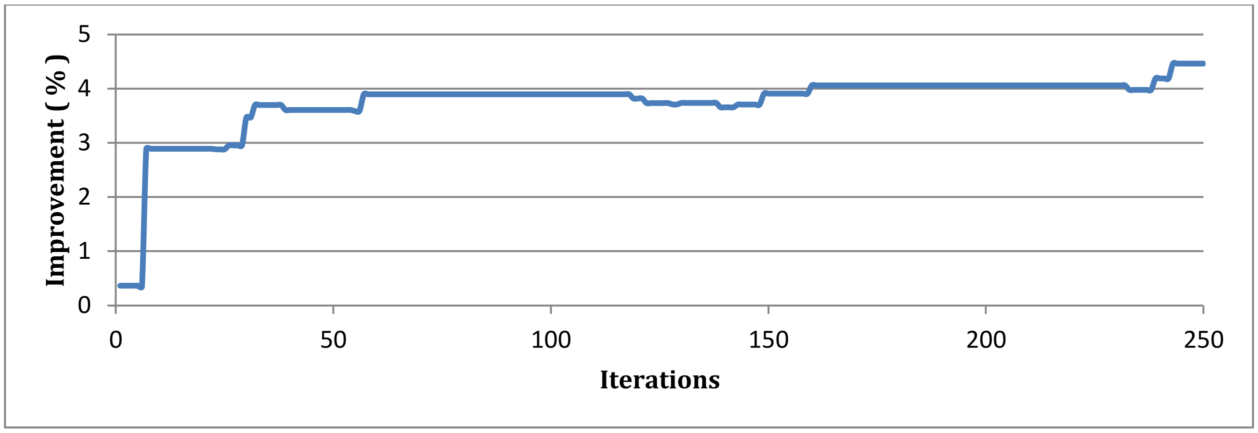


Figure 9 D1T1 scenario; objective function improvement

Table 4 Intersection configurations - Scenario D1T1

Intersection ID	**Configuration**	**Intersection ID**	**Configuration**	**Intersection ID**	**Configuration**
101	2	105	3	109	6
102	3	106	5	110	2
103	1	107	5	111	6
104	4	108	1		

5.4.2 D1T2 scenario

As illustrated in figure 10, 5.2 % improvement is gained with this scenario. In other words, evacuees' total travel time decreases by 5.2 %. A total of 15 intersections are identified as well as the associated configurations through the solution approach to apply crossing elimination as shown in table 5.

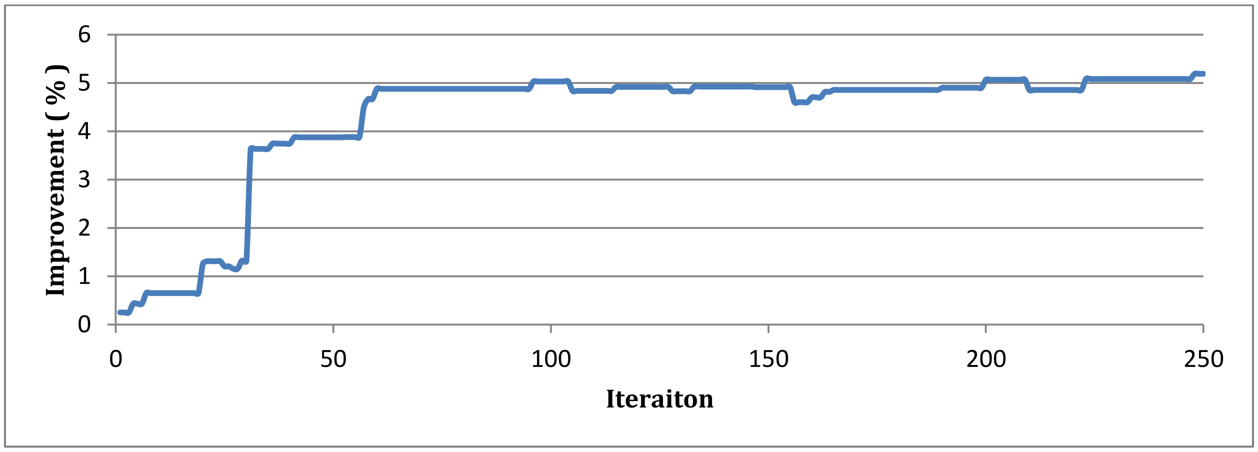


Figure 10 D1T2 Scenario; objective function improvement

Table 5 Intersection configurations - Scenario D1T2

Intersection ID	Configuration	Intersection ID	Configuration	Intersection ID	Configuration
112	3	116	4	107	6
113	4	117	1	121	2
114	1	118	3	122	4
115	3	119	4	110	3
102	2	120	5	123	1

5.4.3 D2T1 scenario

This scenario resulted in a 3% improvement in the objective function as shown in figure 11. In other words, evacuees' total travel time decreases by 3 %. A total of 10 intersections are identified as well as the associated configurations through the solution approach to apply crossing elimination as shown in table 6.

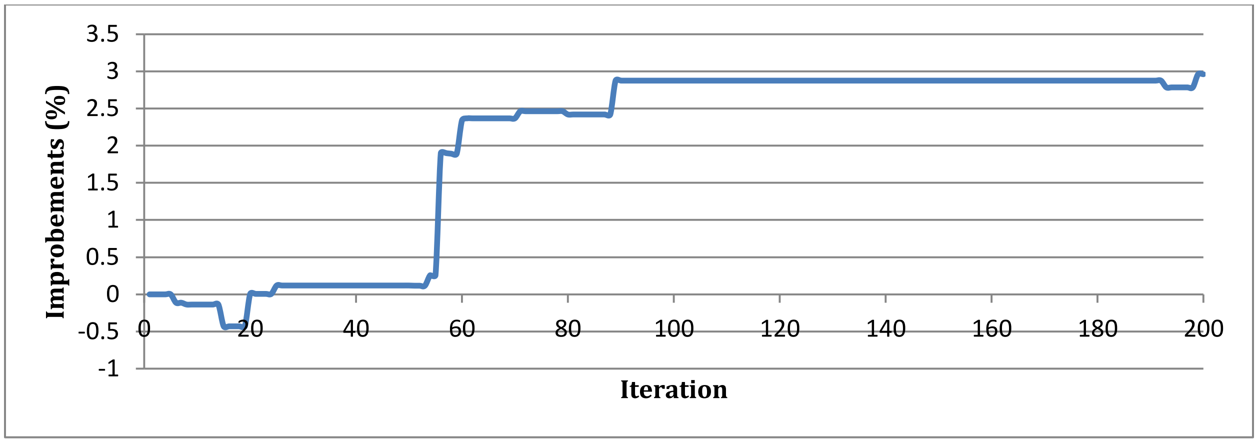


Figure 11 D2T1 Scenario; objective function improvement

Table 6 Intersection configurations - Scenario D1T2

Intersection ID	**Configuration**	**Intersection ID**	**Configuration**	**Intersection ID**	**Configuration**
102	2	118	6	126	2
104	1	122	3	127	4
107	5	124	4		
113	5	125	3		

Scenarios D1T1, D1T2, and D2T1 lead to choosing 27 intersections from the total of 67 as shown in table 7. Also, two intersections are selected in all three scenarios, and are identified as the most critical locations. Moreover, five intersections are selected in two scenarios, that shows they can be critical. In these seven critical cases, different configurations are selected, but configurations 3, 4, 5, and 2 are chosen four, three, three, and three times respectively. However, repeated configurations are not related to a specific intersection. Having the results of more scenarios can help determine the critical intersections more reliably.

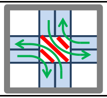
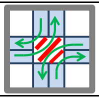
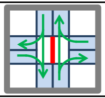
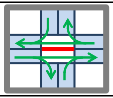
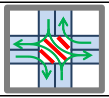
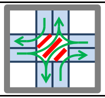
Table 7 intersection selection - critical intersections

Intersection IDs	**Scenarios**			**Number of times intersection is selected**
	D1T1	**D1T2**	**D2T1**	
101	2			1
102	3	2	2	3
103	1			1
104	4		1	2
105	3			1
106	5			1
107	5	6	5	3
108	1			1
109	6			1
110	2	3		2
111	6			1
112		3		1
113		4	5	2
114		1		1

115		3		1
116		4		1
117		1		1
118		3	6	2
119		4		1
120		5		1
121		2		1
122		4	3	2
123		1		1
124			4	1
125			3	1
126			2	1
127			4	1

Configurations 1 and 2 can be grouped as they are exactly the same when rotating one by 90 degrees. Similarly, configurations 3 and 4, and also 5 and 6 are identical and can be grouped as well. Table 8 shows how many times a configuration/group of configurations is selected. The configuration group of 3 and 4 is selected the most (11 times). The reason may be the fact that these configurations are the only ones with allowed through movements. All other configurations force motorists to make turns. The configuration group of 1 and 2 is chosen the most after configurations 3 and 4. Even though they are the most restricted configurations among all, they are selected 9 times. The reason might be the fact that these configurations are the only ones without merging points. In other words, vehicles are not interrupted by other vehicles from other approaches, so they can move more smoothly with less delay.

Table 8 Configuration selection

Configuration number	**Icon**	**Numbers of times used in scenarios:**			**Number of times used total**	**Number of times in similar groups**
		D1T1	**D1T2**	**D2T1**		
1		2	3	1	6	12
2		2	2	2	6	
3		2	4	2	8	15
4		1	4	2	7	
5		2	1	2	5	9
6		2	1	1	4	

5.4.4 Scenario comparison

In all scenarios, crossing elimination improved the situation similarly. Total evacuees' travel time is close in scenarios D1T1 and D2T1, but significantly higher than in D1T2. Higher evacuees' travel time in D1T1 makes sense since in T1 evacuees are loaded in a shorter time which makes the situation more critical as a higher traffic volume is forced into the road network that has a limited capacity. Higher evacuees' travel time in D2T1 also makes sense because in D2 motorists are limited to only choose shelters as opposed to D1 in which they are distributed throughout the network. Other than evacuee travel time, average travel time of all users who completed their trips within the simulation period and number of vehicles that fail to complete their trips within the simulation period are examined as shown in table 9. As can be seen from this table, crossing elimination improves the situation for not only evacuees but also all users who completed their trips. It should be pointed out that not everyone benefits from the strategy. In fact, the average travel time is improved but more vehicles fail to complete their trips within the simulation period after applying the strategy.

Table 9 scenario comparison by different measures

Scenario	**Evacuees travel time (min)**		**Average Travel Times (min)**		**Number of vehicles fail to complete their trips within the simulation period**	
	Do nothing	**Applying crossing elimination**	**Do nothing**	**Applying crossing elimination**	**Do nothing**	**Applying crossing elimination**
D1T1	15,243,248	14,751,555	47.6	45.8	2,244	29,703
D1T2	14,117,642	13,736,720	44.5	42.5	1,064	42,630
D2T1	15,742,615	15,220,891	47.3	46.4	1,672	26,608

Chapter 6 Conclusions and future research

6.1 Conclusions

As one of the recent strategies to apply during evacuation, crossing elimination has positive impacts when adopted by itself and without combining it with other strategies such as contraflow operation which was usually the case throughout the literature. Total of three scenarios are examined. Two of them contributed to around 5 percent improvement in total evacuee travel time, and another scenario resulted in about 3 percent improvement.

A bi-level model is developed to evaluate crossing elimination. Unlike most past studies, evacuees are completely distinguished from shadow evacuees and the background traffic. Total evacuee travel time is considered as the objective function of the upper level problem. The lower level problem which is the traffic assignment problem is solved using DynusT, simulation-based dynamic traffic assignment (DTA) software. Applying the simulation package provides the capability of examining other indications than evacuee travel time as other desired measures can be extracted from the simulation outputs.

Average travel time of all users and the number of vehicles that failed to complete their trips within the simulation period are also assessed. These measures help to understand the situation in a better way as they point out other possible consequences. In this case, average travel time for all users is improved, but more vehicles failed to complete their trips by applying the strategy.

Critical intersections are identified considering those intersections that are selected in both scenarios. Moreover, critical movement configurations are identified by finding those that are selected the most. Configurations without any merge points are selected the most after configurations with some through movements allowed. In other words, through movements and merging points are identified as two factors that have an effect on choosing the configurations.

6.2 Future research

Having several different measures in mind can provide different perspectives of the problem which helps to evaluate the problem in a more holistic way. Including different measures into the problem formulation to create a multi-objective optimization model can be a future work to consider.

More scenarios need to be evaluated so critical intersections and movement configurations can be determined more reliably. Identifying these important intersections and configurations is beneficial especially when the resources are limited and/or when there is not sufficient time to implement the strategy completely.

Considering the huge solution space, the solution method can be improved so it can search the solution space in a smarter way to reach good solutions in a shorter time. For instance, keeping very bad solutions in memory would provide the knowledge to skip those without running the simulation and avoid redundant computations. Also, one-dimensional sensitivity analysis can provide some knowledge of the individual impacts of movement configurations for each intersection. In other words, possible movement configurations of only one intersection can be assessed to understand the benefits that specific intersection can provide. The information obtained from the individual effects can be incorporated into the solution technique for algorithm improvement.

Other movement configurations than what this study considered can be included. Also, other types of intersections than four-legged intersections (i.e. three-legged, five-legged and above) can be evaluated to have a more comprehensive evaluation.

References

Bretschneider, S. and A. Kimms (2011). "A basic mathematical model for evacuation problems in urban areas." (Compendex).

Chen, C.-C. and C.-S. Chou (2008). Incorporating Contra Flow Simulation Into a Transit-Based Emergency Evacuation Plan.

Chen, C.-C. and C.-S. Chou (2009). "Modeling and Performance Assessment of a Transit-Based Evacuation Plan Within a Contraflow Simulation Environment." Transportation Research Record: Journal of the Transportation Research Board(2091): pp 40-50.

Chiu, Y.-C. and P. B. Mirchandani (2008). "Online Behavior-Robust Feedback Information Routing Strategy for Mass Evacuation." IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS 9(2).

Cova, T. J. and J. P. Johnson (2003). "A network flow model for lane-based evacuation routing." Transportation Research, Part A (Policy and Practice) 37A(Copyright 2004, IEE): 579-604.

Edara, P., S. Sharma, et al. (2010). "Development of a large-scale traffic simulation model for hurricane evacuation-methodology and lessons learned." Natural hazards review 11(4): 127-139.

Fonseca, D. J., G. P. Moynihan, et al. (2009). "A simulation tool for hurricane evacuation planning." Model. Simul. Eng. 2009: 1-8.

Friso, K., K. Van Zuilekom, et al. (2009). If things do go wrong: lessons learned concerning traffic management during mass evacuation in case of possible extreme flooding in The Netherlands Conference: ICEM09. Delft.

Friso, K., K. M. van Zuilekom, et al. (2011). Evaluation of the National Concept Traffic Management for mass evacuation in the Netherlands. Networking, Sensing and Control (ICNSC), 2011 IEEE International Conference on.

Ghanipoor-Machiani, S., P. Murray-Tuite, et al. (2013). No-Notice Evacuation Management: Ramp Closures Under Varying Budgets and Demand Scenarios. Transportation Research Board. Washington D.C.

Glover, F. and G. A. Kochenberger (2003). Handbook of metaheuristics, Kluwer academic: New York.

Guterbock, T. M., J. H. Lambert, et al. (2009). NCR Behavioral Survey: Behavioral Aspects of Sheltering and Evacuation Planning for the National Capital Region. Preliminary Results. .

Han, L. D., F. Yuan, et al. (2006). "Global Optimization of Emergency Evacuation Assignments." Interfaces 36(6): 502-513.

Kalafatas, G. and S. Peeta (2009). "Planning for evacuation: Insights from an efficient network design model." Journal of Infrastructure Systems 15(Compendex): 21-30.

Kirkpatrick, S., C. D. Gelatt, et al. (1983). "Optimization by simulated annealing." Science 220(4598).

Lian-bo, D. and S. Feng (2009). The bi-level program and simulated annealing algorithm for the construction sequence of transport network. Second International Conference on Intelligent Computation Technology and Automation.

Liu, Y., G.-L. Chang, et al. (2008). "Corridor-based emergency evacuation system for Washington, D.C. system development and case study." Transportation Research Record(Compendex): 58-67.

Liu, Y. and Z. Luo (2012). "A Bi-Level Model for Planning Signalized and Uninterrupted Flow Intersections in an Evacuation Network." Computer-Aided Civil and Infrastructure Engineering 27(10): 731-747.

McGhee, C. C. and M. C. Grimes (2006). Operational Analysis of the Hampton Roads Hurricane Evacuation Traffic Control Plan. United States: 32p.

Michiels, W., E. Aarts, et al. (2007). Theoretical aspects of local search, Springer: Berlin.

Miller-Hooks, E. and P. Tarnoff (2005). Synopsis of interviews for signal timing for evacuation, Federal Highway Administration

Ming, C., C. Lichun, et al. (2007). "Traffic Signal Timing for Urban Evacuation." Journal of Urban Planning & Development 133(1): 30-42.

Murray-Tuite, P. M., B. B. Park, et al. (2012). EVACUATION TRANSPORTATION ANALYSIS FOR THE NORTHERN VIRGINIA REGION.

Ng, M., J. Park, et al. (2010). "A Hybrid Bilevel Model for the Optimal Shelter Assignment in Emergency Evacuations." Computer-Aided Civil and Infrastructure Engineering 25(Compendex): 547-556.

Patel, R. K. (2006). Improved Managed Lanes Framework for Emergency Management. 85th Annual Meeting of the Transportation Research Board. Washington, D.C., Transportation Research Board.

Pielke, J., R.A. (1998). Forecasting Danger: The Science of Disaster Prediction. Microsoft Encarta Encyclopedia.

Radwan, A. E., A. G. Hobeika, et al. (1985). "A Computer Simulation Model for Rural Network Evacuation Under Natural Disasters." Institute of Transportation Engineers Journal 55(9).

Ren, G., Z. Huang, et al. (2012). "An integrated model for evacuation routing and traffic signal optimization with background demand uncertainty." Journal of advanced transportation.

Romeo F, S.-V. (1991). "A Theoretical Framework for Simulated Annealing." Algorithmica 6: 302-345.

So, S. K. and C. F. Daganzo (2010). "Managing evacuation routes." Transportation Research Part B: Methodological 44(Compendex): 514-520.

Theodoulou, G. and B. Wolshon (2004). "Alternative Methods to Increase the Effectiveness of Freeway Contraflow Evacuation." Transportation Research Record 1865: 48-56.

Theodoulou, G. and B. Wolshon (2004). Modeling and Analyses of Freeway Contraflow to Improve Future Evacuations. 83rd Annual Meeting of the Transportation Research Board. Washington, D.C., Transportation Research Board.

Tuydes, H. (2006). Tabu-Based Heuristic for Optimization of Network Evacuation Contraflow. 85th Annual Meeting of the Transportation Research Board. Washington, D.C., Transportation Research Board.

Tuydes, H. and A. Ziliaskopoulos (2004). Network Re-design to Optimize Evacuation Contraflow. 83rd Annual Meeting of the Transportation Research Board. Washington, D.C., Transportation Research Board.

Tuydes, H. and A. Ziliaskopoulos (2006). Tabu-based heuristic approach for optimization of network evacuation contraflow. Network Modeling 2006, 2001 Wisconsin Avenue NW, Green Building, Washington, DC 20007, United States, National Research Council.

Wolshon, B. (2001). ""One-Way-Out": Contraflow Freeway Operation for Hurricane Evacuation." Natural Hazards Review 2(3): 105-112.

Wolshon, B. C.-M., Alison; Lambert, Laurence (2006). "Louisiana Highway Evacuation Plan for Hurricane Katrina: Proactive Management of a Regional Evacuation." Journal of Transportation Engineering 132(1): 1-10.

Xie, C., D.-Y. Lin, et al. (2010). "A dynamic evacuation network optimization problem with lane reversal and crossing elimination strategies." Transportation Research Part E: Logistics and Transportation Review 46(3): 295-316.

Xie, C. and M. A. Turnquist (2009). "Integrated evacuation network optimization and emergency vehicle assignment." Transportation Research Record(Compendex): 79-90.

Xie, C. and M. A. Turnquist (2011). "Lane-based evacuation network optimization: An integrated Lagrangian relaxation and tabu search approach." Transportation Research Part C: Emerging Technologies 19(Copyright 2011, The Institution of Engineering and Technology): 40-63.

Xie, C., S. T. Waller, et al. (2011). "AN INTERSECTION ORIGIN-DESTINATION FLOW OPTIMIZATION PROBLEM FOR EVACUATION NETWORK DESIGN." TRB.

Appendix A: Source Code

The entire Matlab code is provided below as well as a function written to modify VMS information.

Main code

```
%% Inputs

FiveMileNet= 'E:\Data\Arash\CrossElimination\Crossing Elimination_Final\RUN_beginning\5 ml';

EightMileNet= 'E:\Data\Arash\CrossElimination\Crossing Elimination_Final\RUN_beginning\8 ml';

matlabROOT = 'E:\Data\Arash\CrossElimination\Crossing Elimination_Final\RUN_beginning';

cd(FiveMileNet);

move= dlmread('movement.dat');

fid= fopen('control.dat');

control= textscan(fid,'%n %n %n %n %n %n %n %n %n %n','commentstyle','='); % reading control file:
control file has 3 different main sections

fclose(fid);

for j= 1:10

    cont(:,j)=control{1,j};

end;

id=fopen('network.dat');

network=textscan(id,'%d16 %d16 %*d16 %*d16 %*d16 %*d16 %*d16 %*d16 %*d16 %*d16 %*d16
%d16 %*[^\n]','HeaderLines',821);

fclose(id);

net=cell2mat(network);

intmax=500;
```

```
VMS=zeros(1,7);

%% Identifying 4-leg intersections

numint=0;

numberofnodes=879;

for i=3:numberofnodes

    b=find(move(:,1)==cont(i,1)); %finding node i in all IDs in the first column of movement.dat

    % b=number of outbound links

    c=find(move(:,2)==cont(i,1)); %finding node i in all IDs in the second column of movement.dat

    % c=number of inbound links

    reram1=find(net(:,1)==cont(i,1)); % find row ID of the node ID in Network.dat-outbount link

    reram2=find(net(:,2)==cont(i,1)); % find row ID of the node ID in Network.dat-inbound link

    rampout=net(reram1,3); % column 3 in Network.dat shows link type=link type for outbound link

    rampin=net(reram2,3); % link type for inbound link

    if numel(b)==4 && numel(c)==4 %finding 4-leg approaches that have 4 inbound links and 4 outbound
links

        if sum(ismember(3:4,rampout))==0 && sum(ismember(3:4,rampin))==0 && cont(i,2)~=6 &&
cont(i,2)~=2 % eliminating ramps, 3&4 erpresent ramps, also eliminating control type 6 & 2

            numint=numint+1; % k=number of 4-leg node

            d(numint)=cont(i,1); % storing 4-leg node IDs in d

            %dd(numint)=cont(i,2);

        end;

    end;

end;
```

```
d(d==5506)=[];numint=numint-1;

d(d==5659)=[];numint=numint-1;


d(d==5078)=[];numint=numint-1;

d(d==5151)=[];numint=numint-1;

d(d==5155)=[];numint=numint-1;

d(d==5205)=[];numint=numint-1;

d(d==5660)=[];numint=numint-1;

d(d==6050)=[];numint=numint-1;

d(d==6056)=[];numint=numint-1;

d(d==6074)=[];numint=numint-1;


d(d==5544)=[];numint=numint-1;


d(d==772)=[];numint=numint-1;

d(d==5099)=[];numint=numint-1;

d(d==5190)=[];numint=numint-1;

d(d==5120)=[];numint=numint-1;

d(d==5215)=[];numint=numint-1;

d(d==5163)=[];numint=numint-1;

d(d==5173)=[];numint=numint-1;

d(d==5237)=[];numint=numint-1;
```

```
d(d==5243)=[];numint=numint-1;

d(d==5278)=[];numint=numint-1;

d(d==5313)=[];numint=numint-1;

d(d==5331)=[];numint=numint-1;

d(d==5349)=[];numint=numint-1;

d(d==5558)=[];numint=numint-1;

d(d==5610)=[];numint=numint-1;

d(d==5619)=[];numint=numint-1;

d(d==5651)=[];numint=numint-1;

d(d==6048)=[];numint=numint-1;

d(d==6053)=[];numint=numint-1;


%% reading input files from main DynusT root

clear cont

clear move

cd(EightMileNet);

fid_result = fopen('results.dat','w');

move= dlmread('movement.dat');

fid=fopen('control.dat');

control=textscan(fid,'%n %n %n %n %n %n %n %n %n %n','commentstyle','='); % reading control file:
control file has 3 different main sections

fclose(fid);

for j=1:10
```

```
    cont(:,j)=control{1,j};

end;

cont_orig=cont;

cont_modified=cont;

%% Simulated Annealing

%Determining the parameters needed for the algorithm

Initial_t=0.1; Final_t=0.01; Num_iterations=50;

T=Initial_t; %current temperature


%% DynusT RUN


!DynusTv3bx64.exe&

stop=0;

RUNexe=fullfile(EightMileNet,'Executing'); % there is file named Executing in the DynusT folder, when
DynusT is running it has a value of 1,

% but once DynusT has completed its run this file changes to an empty file

disp('DynusT is RUNNING... ');


while stop==0

    fid=fopen(RUNexe,'r');

    if fid==-1 % this means that the file(Executing) is empty

        disp('DynusT has completed its "initial" run.');

        stop=1;
    else

        fclose(fid);

        pause(2); % 2 seconds wait before checking the execution file again

    end

end

% getting the objective function of the initial solution

%% initial solution

% without any restriction



impzone=dlmread('impzone.txt'); %read impacted zones



fid=fopen('VehTrajectory.dat'); %read vehicle trajectory from DynusT folder

vehODT=textscan(fid,'%s %*s %*s %*s %*s %*s %d16 %*s %*s %*s %*s %*s %*s %*s %*s %*s
%*s %*s %d16 %*s %*s %*s %n %*[^\n]','HeaderLines',4);

fclose(fid); %format: *->skip the field/ first field is read as s(str)/ 7th

% & 9th are read as d16(long integer)/ 23rd is read as

% n(floating point num)/ %*[^\n]->skip the rest/

%'HeaderLines',4 ->the first 4 lines are ignored



initialOBJECTIVE=0; % initial value for objective function

rowID=find(ismember(vehODT{1,1}, 'Veh')==1); %finding the rows that start with Veh

for i=1:numel(rowID)

    %display(i);
    if ismember(vehODT{1,2}(rowID(i)),impzone)==1 && vehODT{1,3}(rowID(i))>=60

        initialOBJECTIVE=initialOBJECTIVE+vehODT{1,4}(rowID(i));

        %for each vehicle, if the origon is an internal zone and departure

        %time is greater than 60 min, add its total travel time (OD

        %travel time) to the objective function variable

    end

end

current=initialOBJECTIVE;

fprintf(fid_result, '%s%f\n','initial objective =', initialOBJECTIVE);

fprintf(fid_result, '%s\n', '*********');

stopcount=0;

disp(['initial objective function: ' num2str(initialOBJECTIVE)]);

%%

tempStage=0;

h=0;

budget=1;

while T>=Final_t

    tempStage=tempStage+1;

    disp(['**********Temperature: ' num2str(T) ' ***Stage: ' num2str(tempStage)
'*****************************************************************************']);

    acc=0;

    accwors=0;

    wors=0;
    objfun_startT=current;

    for ite=1:Num_iterations

        %% defining configurations (modifying control.dat)

        norm=0;

        while norm==0

            e=ceil(numint*rand); % a random number between 0 and numint - picking a node arbitrarily from
all 4-leg intersections

            f=find(move(:,2)==d(e)); % finding the randomly selected node in the second column of
movment.dat

            if budget==0

                if sum(ismember(d(e),config(1,config(3,:)==1)))==0 %
sum(ismember(d(e),config(1,find(config(3,:)==1))))

                    e_add=ceil(numel(config(1,config(3,:)==1))*rand);

                    dd=config(1,config(3,:)==1); % dd: the first row of config which include 4-leg signalized
IDs, but only those for which configurations were assigned

                    addi=[];

                    numberofnodes2=1523;


f_node=find(cont_modified(numberofnodes2+3:size(cont_modified,1),1)==dd(e_add))+numberofnodes2
+2;

                    bref=1;

                    while bref<=numel(f_node)

                        if sum(isnan(cont_modified(f_node(bref),:)))~=0

                            f_node(bref,:)=[];
                            bref=bref-1;
                        end;
                        bref=bref+1;
                    end;
                    for z=1:numel(f_node)

f_node=find(cont_modified(numberofnodes2+3:size(cont_modified,1),1)==dd(e_add))+numberofnodes2
+2;

f_node_orig=find(cont_orig(numberofnodes2+3:size(cont_orig,1),1)==dd(e_add))+numberofnodes2+2;
                        bref=1;
                        while bref<=numel(f_node)
                            if sum(isnan(cont_modified(f_node(bref),:)))~=0
                                f_node(bref,:)=[];
                                bref=bref-1;
                            end;
                            bref=bref+1;
                        end;
                        bref=1;
                        while bref<=numel(f_node_orig)
                            if sum(isnan(cont_orig(f_node_orig(bref),:)))~=0
                                f_node_orig(bref,:)=[];
                                bref=bref-1;
                            end;

                            bref=bref+1;

                        end;

                        for y=1:cont_modified(f_node(z),6)

                            cont_modified(f_node(z)+1,:)=[];

                        end;

                        cont_modified(f_node(z),:)=cont_orig(f_node_orig(z),:);

                        cont_modified(f_node(z),5)=1;

                        k=1;

                        for v=1:cont_orig(f_node_orig(z),6)

                            addi(v,:)=cont_orig(f_node_orig(z)+v,:);

                            %addi(v,numel(addi(v,:))+1:10)=nan;

                        end;

                        row_indic=f_node(z);


cont_modified=[cont_modified(1:row_indic,:);addi;cont_modified(row_indic+1:size(cont_modified,1),:)]
;

                    end;

                  end;

                end;

                fs(1)=f(1); % fs: the same as f, but sorted in this way:fs=(aa,bb,cc,dd)

                % aa= the first To_ID row number in movement.dat found in line 13
                % bb= Downstream node from To_ID representing left-turn movement(row number in movement.dat, not the actual node ID)

                % cc= Downstream node from To_ID representing through movement(row number in movement.dat, not the actual node ID)

                % dd= Downstream node from To_ID representing right-turn movement(row number in movement.dat, not the actual node ID)

                l=move(f(1),3); % finding the actual Downstream node from To_ID representing left-turn movement

                t=move(f(1),4); % finding the actual Downstream node from To_ID representing through movement

                r=move(f(1),5); % finding the actual Downstream node from To_ID representing right-turn movement

                for j=2:4 % creating fs

                  if move(f(j),1)==l

                    fs(2)=f(j);

                  elseif move(f(j),1)==t

                    fs(3)=f(j);

                  elseif move(f(j),1)==r

                    fs(4)=f(j);

                  end;

                end;

                [mem,index]=ismember(fs,f);

                zer=ismember([1,2,3,4],index);

                fs(mem==0)=f(zer==0); % fs(find(mem==0))=f(find(zer==0));
```

```
            if numel(fs)==4
                norm=1;
            end;
        end;
        %%
        ee=ceil(7*rand); % a random number between 1 and 7
        cont_pre=cont_modified;
        numberofnodes2=1523;
        switch ee
            case 1 %configuration 1

                addi=[];

f_node=find(cont_modified(numberofnodes2+3:size(cont_modified,1),1)==d(e))+numberofnodes2+2;
                bref=1;
                while bref<=numel(f_node)
                    if sum(isnan(cont_modified(f_node(bref),:)))~=0
                        f_node(bref,:)=[];
                        bref=bref-1;
                    end;
                    bref=bref+1;
                end;
                for z=1:numel(f_node)
```

```
f_node=find(cont_modified(numberofnodes2+3:size(cont_modified,1),1)==d(e))+numberofnodes2+2;

          bref=1;

          while bref<=numel(f_node)

            if sum(isnan(cont_modified(f_node(bref),:)))~=0

              f_node(bref,:)=[];

              bref=bref-1;

            end;

            bref=bref+1;

          end;

          for y=1:cont_modified(f_node(z),6)

            cont_modified(f_node(z)+1,:)=[];

          end;

          cont_modified(f_node(z),5)=1;

          cont_modified(f_node(z),6)=4;

          for a=1:4

            cont_modified(f_node(z),a+6)=move(f(a),1);

          end;


          k=1;

          for v=1:4

            addi(k,1)=move(f(v),1);

            addi(k,2)=move(f(v),2);
addi(k,3)=z;
addi(k,4)=1;
switch v
    case 1
        addi(k,5)=move(fs(4),1);
    case 2
        switch find(fs==f(v))
            case 2 % 2:left turn
                addi(k,5)=move(fs(3),1);
            case 3 % 3:through
                addi(k,5)=move(fs(2),1);
            case 4 % 4:right turn
                addi(k,5)=move(fs(1),1);
        end;
    case 3
        switch find(fs==f(v))
            case 2
                addi(k,5)=move(fs(3),1);
            case 3
                addi(k,5)=move(fs(2),1);
            case 4
                addi(k,5)=move(fs(1),1);
```

```
                    end;
                  case 4
                    switch find(fs==f(v))
                      case 2
                        addi(k,5)=move(fs(3),1);
                      case 3
                        addi(k,5)=move(fs(2),1);
                      case 4
                        addi(k,5)=move(fs(1),1);
                    end;
                end;
                addi(k,6:10)=nan;
                k=k+1;
              end;
              row_indic=f_node(z);

cont_modified=[cont_modified(1:row_indic,:);addi;cont_modified(row_indic+1:size(cont_modified,1),:)]
;
            end;

          case 2 %configuration 2

            addi=[];
```

```
f_node=find(cont_modified(numberofnodes2+3:size(cont_modified,1),1)==d(e))+numberofnodes2+2;
            bref=1;
            while bref<=numel(f_node)
               if sum(isnan(cont_modified(f_node(bref),:)))~=0
                  f_node(bref,:)=[];
                  bref=bref-1;
               end;
               bref=bref+1;
            end;
            for z=1:numel(f_node)

f_node=find(cont_modified(numberofnodes2+3:size(cont_modified,1),1)==d(e))+numberofnodes2+2;
               bref=1;
               while bref<=numel(f_node)
                  if sum(isnan(cont_modified(f_node(bref),:)))~=0
                     f_node(bref,:)=[];
                     bref=bref-1;
                  end;
                  bref=bref+1;
               end;
               for y=1:cont_modified(f_node(z),6)
                  cont_modified(f_node(z)+1,:)=[];
```

```
end;
cont_modified(f_node(z),5)=1;
cont_modified(f_node(z),6)=4;
for a=1:4
    cont_modified(f_node(z),a+6)=move(f(a),1);
end;

k=1;
for v=1:4
    addi(k,1)=move(f(v),1);
    addi(k,2)=move(f(v),2);
    addi(k,3)=z;
    addi(k,4)=1;
    switch v
        case 1
            addi(k,5)=move(fs(2),1);
        case 2
            switch find(fs==f(v))
                case 2 % 2:left turn
                    addi(k,5)=move(fs(1),1);
                case 3 % 3:through
                    addi(k,5)=move(fs(4),1);
                case 4 % 4:right turn
                    addi(k,5)=move(fs(3),1);
            end;
        case 3
            switch find(fs==f(v))
                case 2
                    addi(k,5)=move(fs(1),1);
                case 3
                    addi(k,5)=move(fs(4),1);
                case 4
                    addi(k,5)=move(fs(3),1);
            end;
        case 4
            switch find(fs==f(v))
                case 2
                    addi(k,5)=move(fs(1),1);
                case 3
                    addi(k,5)=move(fs(4),1);
                case 4
                    addi(k,5)=move(fs(3),1);
            end;
    end;
```

```
                addi(k,6:10)=nan;

                k=k+1;

            end;

            row_indic=f_node(z);


cont_modified=[cont_modified(1:row_indic,:);addi;cont_modified(row_indic+1:size(cont_modified,1),:)]
;

        end;


      case 3 %configuration 3


        addi=[];


f_node=find(cont_modified(numberofnodes2+3:size(cont_modified,1),1)==d(e))+numberofnodes2+2;

        bref=1;

        while bref<=numel(f_node)

            if sum(isnan(cont_modified(f_node(bref),:)))~=0

                f_node(bref,:)=[];

                bref=bref-1;

            end;

            bref=bref+1;

        end;

        for z=1:numel(f_node)
```

```
f_node=find(cont_modified(numberofnodes2+3:size(cont_modified,1),1)==d(e))+numberofnodes2+2;
          bref=1;
          while bref<=numel(f_node)
            if sum(isnan(cont_modified(f_node(bref),:)))~=0
              f_node(bref,:)=[];
              bref=bref-1;
            end;
            bref=bref+1;
          end;
          for y=1:cont_modified(f_node(z),6)
            cont_modified(f_node(z)+1,:)=[];
          end;
          cont_modified(f_node(z),5)=1;
          cont_modified(f_node(z),6)=4;
          for a=1:4
            cont_modified(f_node(z),a+6)=move(f(a),1);
          end;


          k=1;
          for v=1:4
            addi(k,1)=move(f(v),1);
            addi(k,2)=move(f(v),2);
addi(k,3)=z;
switch v
    case 1
        addi(k,5)=move(fs(4),1);
        addi(k,4)=1;addi(k,6:10)=nan;
    case 2
        switch find(fs==f(v))
            case 2 % 2:left turn
                addi(k,5)=move(fs(1),1);
                addi(k,6)=move(fs(4),1);
                addi(k,4)=2;addi(k,7:10)=nan;
            case 3 % 3:through
                addi(k,5)=move(fs(2),1);
                addi(k,4)=1;addi(k,6:10)=nan;
            case 4 % 4:right turn
                addi(k,5)=move(fs(2),1);
                addi(k,6)=move(fs(3),1);
                addi(k,4)=2;addi(k,7:10)=nan;
        end;
    case 3
        switch find(fs==f(v))
            case 2
        addi(k,5)=move(fs(1),1);

        addi(k,6)=move(fs(4),1);

        addi(k,4)=2;addi(k,7:10)=nan;

      case 3

        addi(k,5)=move(fs(2),1);

        addi(k,4)=1;addi(k,6:10)=nan;

      case 4

        addi(k,5)=move(fs(2),1);

        addi(k,6)=move(fs(3),1);

        addi(k,4)=2;addi(k,7:10)=nan;

    end;

  case 4

    switch find(fs==f(v))

      case 2

        addi(k,5)=move(fs(1),1);

        addi(k,6)=move(fs(4),1);

        addi(k,4)=2;addi(k,7:10)=nan;

      case 3

        addi(k,5)=move(fs(2),1);

        addi(k,4)=1;addi(k,6:10)=nan;

      case 4

        addi(k,5)=move(fs(2),1);
```

```
                    addi(k,6)=move(fs(3),1);

                    addi(k,4)=2;addi(k,7:10)=nan;

                end;

            end;

            k=k+1;

          end;

          row_indic=f_node(z);

cont_modified=[cont_modified(1:row_indic,:);addi;cont_modified(row_indic+1:size(cont_modified,1),:)]
;

        end;


      case 4 % configuration 4


        addi=[];

f_node=find(cont_modified(numberofnodes2+3:size(cont_modified,1),1)==d(e))+numberofnodes2+2;

        bref=1;

        while bref<=numel(f_node)

          if sum(isnan(cont_modified(f_node(bref),:)))~=0

            f_node(bref,:)=[];

            bref=bref-1;

          end;
```

```
            bref=bref+1;
          end;
          for z=1:numel(f_node)

f_node=find(cont_modified(numberofnodes2+3:size(cont_modified,1),1)==d(e))+numberofnodes2+2;
            bref=1;
            while bref<=numel(f_node)
              if sum(isnan(cont_modified(f_node(bref),:)))~=0
                f_node(bref,:)=[];
                bref=bref-1;
              end;
              bref=bref+1;
            end;
            for y=1:cont_modified(f_node(z),6)
              cont_modified(f_node(z)+1,:)=[];
            end;
            cont_modified(f_node(z),5)=1;
            cont_modified(f_node(z),6)=4;
            for a=1:4
              cont_modified(f_node(z),a+6)=move(f(a),1);
            end;


            k=1;
```

```
for v=1:4
    addi(k,1)=move(f(v),1);
    addi(k,2)=move(f(v),2);
    addi(k,3)=z;
    switch v
        case 1
            addi(k,5)=move(fs(3),1);
            addi(k,6)=move(fs(4),1);
            addi(k,4)=2;addi(k,7:10)=nan;
        case 2
            switch find(fs==f(v))
                case 2 % 2:left turn
                    addi(k,5)=move(fs(1),1);
                    addi(k,4)=1;addi(k,6:10)=nan;
                case 3 % 3:through
                    addi(k,5)=move(fs(1),1);
                    addi(k,6)=move(fs(2),1);
                    addi(k,4)=2;addi(k,7:10)=nan;
                case 4 % 4:right turn
                    addi(k,5)=move(fs(3),1);
                    addi(k,4)=1;addi(k,6:10)=nan;
            end;
```

```
case 3
    switch find(fs==f(v))
        case 2 % 2:left turn
            addi(k,5)=move(fs(1),1);
            addi(k,4)=1;addi(k,6:10)=nan;
        case 3 % 3:through
            addi(k,5)=move(fs(1),1);
            addi(k,6)=move(fs(2),1);
            addi(k,4)=2;addi(k,7:10)=nan;
        case 4 % 4:right turn
            addi(k,5)=move(fs(3),1);
            addi(k,4)=1;addi(k,6:10)=nan;
    end;
case 4
    switch find(fs==f(v))
        case 2 % 2:left turn
            addi(k,5)=move(fs(1),1);
            addi(k,4)=1;addi(k,6:10)=nan;
        case 3 % 3:through
            addi(k,5)=move(fs(1),1);
            addi(k,6)=move(fs(2),1);
            addi(k,4)=2;addi(k,7:10)=nan;
```

```
                    case 4 % 4:right turn
                        addi(k,5)=move(fs(3),1);
                        addi(k,4)=1;addi(k,6:10)=nan;
                  end;
          end;
          k=k+1;
      end;
      row_indic=f_node(z);
cont_modified=[cont_modified(1:row_indic,:);addi;cont_modified(row_indic+1:size(cont_modified,1),:)]
;
    end;

  case 5 % comfiguration 5

    addi=[];
f_node=find(cont_modified(numberofnodes2+3:size(cont_modified,1),1)==d(e))+numberofnodes2+2;
    bref=1;
    while bref<=numel(f_node)
      if sum(isnan(cont_modified(f_node(bref),:)))~=0
        f_node(bref,:)=[];
        bref=bref-1;
```

```
                end;
                bref=bref+1;
            end;
            for z=1:numel(f_node)

f_node=find(cont_modified(numberofnodes2+3:size(cont_modified,1),1)==d(e))+numberofnodes2+2;
                bref=1;
                while bref<=numel(f_node)
                    if sum(isnan(cont_modified(f_node(bref),:)))~=0
                        f_node(bref,:)=[];
                        bref=bref-1;
                    end;
                    bref=bref+1;
                end;
                for y=1:cont_modified(f_node(z),6)
                    cont_modified(f_node(z)+1,:)=[];
                end;
                cont_modified(f_node(z),5)=1;
                cont_modified(f_node(z),6)=4;
                for a=1:4
                    cont_modified(f_node(z),a+6)=move(f(a),1);
                end;
```

```
k=1;

for v=1:4

    addi(k,1)=move(f(v),1);

    addi(k,2)=move(f(v),2);

    addi(k,3)=z;

    switch v

        case 1

            addi(k,5)=move(fs(4),1);

            addi(k,4)=1;addi(k,6:10)=nan;

        case 2

            switch find(fs==f(v))

                case 2 % 2:left turn

                    addi(k,5)=move(fs(3),1);

                    addi(k,6)=move(fs(1),1);

                    addi(k,4)=2;addi(k,7:10)=nan;

                case 3 % 3:through

                    addi(k,5)=move(fs(2),1);

                    addi(k,4)=1;addi(k,6:10)=nan;

                case 4 % 4:right turn

                    addi(k,5)=move(fs(1),1);

                    addi(k,6)=move(fs(3),1);

                    addi(k,4)=2;addi(k,7:10)=nan;
            end;
        case 3
            switch find(fs==f(v))
                case 2 % 2:left turn
                    addi(k,5)=move(fs(3),1);
                    addi(k,6)=move(fs(1),1);
                    addi(k,4)=2;addi(k,7:10)=nan;
                case 3 % 3:through
                    addi(k,5)=move(fs(2),1);
                    addi(k,4)=1;addi(k,6:10)=nan;
                case 4 % 4:right turn
                    addi(k,5)=move(fs(1),1);
                    addi(k,6)=move(fs(3),1);
                    addi(k,4)=2;addi(k,7:10)=nan;
            end;
        case 4
            switch find(fs==f(v))
                case 2 % 2:left turn
                    addi(k,5)=move(fs(3),1);
                    addi(k,6)=move(fs(1),1);
                    addi(k,4)=2;addi(k,7:10)=nan;
                case 3 % 3:through
```

```
                    addi(k,5)=move(fs(2),1);

                    addi(k,4)=1;addi(k,6:10)=nan;

                  case 4 % 4:right turn

                    addi(k,5)=move(fs(1),1);

                    addi(k,6)=move(fs(3),1);

                    addi(k,4)=2;addi(k,7:10)=nan;

                end;

            end;

            k=k+1;

          end;

          row_indic=f_node(z);

cont_modified=[cont_modified(1:row_indic,:);addi;cont_modified(row_indic+1:size(cont_modified,1),:)]
;

        end;


      case 6 % configuration 6


        addi=[];


f_node=find(cont_modified(numberofnodes2+3:size(cont_modified,1),1)==d(e))+numberofnodes2+2;

        bref=1;

        while bref<=numel(f_node)
```

```
            if sum(isnan(cont_modified(f_node(bref),:)))~=0
                f_node(bref,:)=[];
                bref=bref-1;
            end;
            bref=bref+1;
        end;
        for z=1:numel(f_node)

f_node=find(cont_modified(numberofnodes2+3:size(cont_modified,1),1)==d(e))+numberofnodes2+2;
            bref=1;
            while bref<=numel(f_node)
                if sum(isnan(cont_modified(f_node(bref),:)))~=0
                    f_node(bref,:)=[];
                    bref=bref-1;
                end;
                bref=bref+1;
            end;
            for y=1:cont_modified(f_node(z),6)
                cont_modified(f_node(z)+1,:)=[];
            end;
            cont_modified(f_node(z),5)=1;
            cont_modified(f_node(z),6)=4;
            for a=1:4
```

```
    cont_modified(f_node(z),a+6)=move(f(a),1);
end;


k=1;
for v=1:4
    addi(k,1)=move(f(v),1);
    addi(k,2)=move(f(v),2);
    addi(k,3)=z;
    switch v
        case 1
            addi(k,5)=move(fs(2),1);
            addi(k,6)=move(fs(4),1);
            addi(k,4)=2;addi(k,7:10)=nan;
        case 2
            switch find(fs==f(v))
                case 2 % 2:left turn
                    addi(k,5)=move(fs(1),1);
                    addi(k,4)=1;addi(k,6:10)=nan;
                case 3 % 3:through
                    addi(k,5)=move(fs(4),1);
                    addi(k,6)=move(fs(2),1);
                    addi(k,4)=2;addi(k,7:10)=nan;
                case 4 % 4:right turn
                    addi(k,5)=move(fs(3),1);
                    addi(k,4)=1;addi(k,6:10)=nan;
            end;
        case 3
            switch find(fs==f(v))
                case 2 % 2:left turn
                    addi(k,5)=move(fs(1),1);
                    addi(k,4)=1;addi(k,6:10)=nan;
                case 3 % 3:through
                    addi(k,5)=move(fs(4),1);
                    addi(k,6)=move(fs(2),1);
                    addi(k,4)=2;addi(k,7:10)=nan;
                case 4 % 4:right turn
                    addi(k,5)=move(fs(3),1);
                    addi(k,4)=1;addi(k,6:10)=nan;
            end;
        case 4
            switch find(fs==f(v))
                case 2 % 2:left turn
                    addi(k,5)=move(fs(1),1);
                    addi(k,4)=1;addi(k,6:10)=nan;
```

```
                    case 3 % 3:through
                        addi(k,5)=move(fs(4),1);
                        addi(k,6)=move(fs(2),1);
                        addi(k,4)=2;addi(k,7:10)=nan;
                    case 4 % 4:right turn
                        addi(k,5)=move(fs(3),1);
                        addi(k,4)=1;addi(k,6:10)=nan;
                end;
            end;
            k=k+1;
        end;
        row_indic=f_node(z);

cont_modified=[cont_modified(1:row_indic,:);addi;cont_modified(row_indic+1:size(cont_modified,1),:)]
;
    end;


  case 7 % configuration 7 going back to the original signal configuration


    % addi=[]; moved to 726

f_node=find(cont_modified(numberofnodes2+3:size(cont_modified,1),1)==d(e))+numberofnodes2+2;
    bref=1;
```

```
            while bref<=numel(f_node)
                if sum(isnan(cont_modified(f_node(bref),:)))~=0
                    f_node(bref,:)=[];
                    bref=bref-1;
                end;
                bref=bref+1;
            end;
            for z=1:numel(f_node)
                addi=[];

f_node=find(cont_modified(numberofnodes2+3:size(cont_modified,1),1)==d(e))+numberofnodes2+2;

f_node_orig=find(cont_orig(numberofnodes2+3:size(cont_orig,1),1)==d(e))+numberofnodes2+2;
                bref=1;
                while bref<=numel(f_node)
                    if sum(isnan(cont_modified(f_node(bref),:)))~=0
                        f_node(bref,:)=[];
                        bref=bref-1;
                    end;
                    bref=bref+1;
                end;
                bref=1;
                while bref<=numel(f_node_orig)
```

```
                    if sum(isnan(cont_orig(f_node_orig(bref),:)))~=0

                        f_node_orig(bref,:)=[];

                        bref=bref-1;

                    end;

                    bref=bref+1;

                end;

                for y=1:cont_modified(f_node(z),6)

                    cont_modified(f_node(z)+1,:)=[];

                end;

                cont_modified(f_node(z),:)=cont_orig(f_node_orig(z),:);

                cont_modified(f_node(z),5)=1;

                k=1;

                for v=1:cont_orig(f_node_orig(z),6)

                    addi(v,:)=cont_orig(f_node_orig(z)+v,:);

                    %addi(v,numel(addi(v,:))+1:10)=nan;

                end;

                row_indic=f_node(z);


cont_modified=[cont_modified(1:row_indic,:);addi;cont_modified(row_indic+1:size(cont_modified,1),:)]
;

            end;



        end;
```

```
%% generating VMS

if acc>=1

   if sum(ismember(d(e),config(1,config(3,:)==1)))==0

      cd(matlabROOT);

      nodeID=d(e);

      VMS_pre=VMS;

      VMS=vmsFUN(VMS,nodeID,move);

   end

end

%% writing VMS

cd(EightMileNet);

fid = fopen('vms.dat','w');

fprintf(fid,'%0.0f\n%*10d %*10d %*10d %*10d %*10d %*10d',VMS(1,:));

for i=2:size(VMS,1)

   if numel(nonzeros(VMS(i,:)))==1

      fprintf(fid,'%7.0f\n%*10d %*10d %*10d %*10d %*10d %*10d',VMS(i,:));

   else

      fprintf(fid,'%2.0f%7.0f%7.0f%4.0f%4.0f%12.0f%12.0f\n',VMS(i,:));

   end

end

fclose(fid);

%% writing the modified control.dat
```

```
fid = fopen('control.dat','w+');

cont_modified(1,1)=2;

line1 = cont_modified(1,~isnan(cont_modified(1,:)));

fprintf(fid,['%3d','\n'],line1);

cont_modified(2,2)=70;

line2 = cont_modified(2,~isnan(cont_modified(2,:)));

fprintf(fid,['%10.2f','%10.2f','\n'],line2);

fclose(fid);

%signal plan 1

fid=fopen('control.dat','a');

for idx=3:numberofnodes2+2

    line3 = cont_orig(idx,~isnan(cont_orig(idx,:)));  % creates the line of data without NaNs

    fprintf(fid,['%7d','%2d','%2d','%4d','\n'],line3);

end;

fclose(fid);

%

sp=0;

fid=fopen('control.dat','a');

for idx=numberofnodes2+3:size(cont_orig,1)

    if sum(isnan(cont_orig(idx,:)))==7 && sp==0

        fprintf(fid,'=======Two Way Stop Signs/Yield Signs Below =======\n');

        sp=1;
    end;
    line45 = cont_orig(idx,~isnan(cont_orig(idx,:)));  % creates the line of data without NaNs
    fprintf(fid,[repmat('%7d',1,length(line45)),'\n'],line45);
end;
fprintf(fid,'\n'); % a blank line between two plans
fclose(fid);


% signal plan 2
fid=fopen('control.dat','a');
for idx=3:numberofnodes2+2
    line3 = cont_modified(idx,~isnan(cont_modified(idx,:)));  % creates the line of data without NaNs
    fprintf(fid,['%7d','%2d','%2d','%4d','\n'],line3);
end;
fclose(fid);
%
sp=0;
fid=fopen('control.dat','a');
for idx=numberofnodes2+3:size(cont_modified,1)
    if sum(isnan(cont_modified(idx,:)))==7 && sp==0
        fprintf(fid,'=======Two Way Stop Signs/Yield Signs Below =======\n');
        sp=1;
    end;
```

```
            line45 = cont_modified(idx,~isnan(cont_modified(idx,:)));  % creates the line of data without
NaNs

            fprintf(fid,[repmat('%7d',1,length(line45)),'\n'],line45);

        end;

        fprintf(fid,'================\n');

        fclose(fid);



        %% new solution

        % new run

        !DynusTv3bx64.exe&

        stop=0;

        RUNexe=fullfile(EightMileNet,'Executing'); % there is file named Executing in the DynusT folder,
when DynusT is running it has a value of 1,

        % but once DynusT has completed its run this file changes to an empty file

        disp('------------------DynusT is RUNNING............ ');



        while stop==0

            fid=fopen(RUNexe,'r');

            if fid==-1 % this means that the file(Executing) is empty

                disp('DynusT has completed its run.');

                stop=1;

            else

                fclose(fid);
                pause(2); % 20 seconds wait before checking the execution file again
            end
        end
        % getting the objective function of the new solution
        fid=fopen('VehTrajectory.dat'); %read vehicle trajectory from DynusT folder
        vehODT=textscan(fid,'%s %*s %*s %*s %*s %*s %d16 %*s %*s %*s %*s %*s %*s %*s %*s
%*s %*s %*s %d16 %*s %*s %*s %n %*[^\n]','HeaderLines',5);
        fclose(fid); %format: *->skip the field/ first field is read as s(str)/ 7th
        % & 9th are read as d16(long integer)/ 23rd is read as
        % n(floating point num)/ %*[^\n]->skip the rest/
        %'HeaderLines',4 ->the first 4 lines are ignored


        new=0; % initial value for objective function
        rowID=find(ismember(vehODT{1,1}, 'Veh')==1); %finding the rows that start with Veh
        for i=1:numel(rowID)
            if ismember(vehODT{1,2}(rowID(i)),impzone)==1 && vehODT{1,3}(rowID(i))>=60
                new=new+vehODT{1,4}(rowID(i));
                %for each vehicle, if the origon is an internal zone and departure
                %time is greater than 60 min, add its total travel time (OD
                %travel time) to the objective function variable
            end
        end
        disp(['iteration: ' num2str(ite) ',  Temp: ' num2str(T) ',  Stage: ' num2str(tempStage)]);
        disp(['new objective function: ' num2str(new)]);

        % comparing the new objective function and the current objective

        % function/ % accepting the new soluition or not?

        relativDiff=(new-current)/((new+current)/2)*100;



        % writing results.dat

        fprintf(fid_result, '%f\n', new);

        fprintf(fid_result, '%s%f\n','iteration=', ite);

        fprintf(fid_result, '%s%f%s%f\n','T=', T,'stage=', tempStage);



        if new<current

            display('better-Accepted');

            % writing results.dat

            fprintf(fid_result, '%s\n', 'better-Accepted');



            cont_final=cont_modified;

            current=new;

            acc=acc+1;

            config(1,e)=d(e);

            config(2,e)=ee;

            if ee~=7

                config(3,e)=1;
            else
                config(3,e)=0;
            end;
        elseif exp(-(relativDiff)/T)>rand
            display('worse-Accepted');
            % writing results.dat
            fprintf(fid_result, '%s\n', 'worse-Accepted');

            cont_final=cont_modified;
            current=new;
            acc=acc+1;
            accwors=accwors+1;
            wors=wors+1;
            config(1,e)=d(e);
            config(2,e)=ee;
            if ee~=7
                config(3,e)=1;
            else
                config(3,e)=0;
            end
        else
            display('worse-NOT Accepted');
```

```
            % writing results.dat
            fprintf(fid_result, '%s\n', 'worse-NOT Accepted');

            cont_final=cont_pre;
            cont_modified=cont_pre;
            if acc >= 1
                VMS=VMS_pre;
            end
            wors=wors+1;
            config(1,e)=d(e);
            config(2,e)=0;
            config(3,e)=0;
        end;
        if sum(config(3,:))==intmax
            disp('!!!!!!!!!!!============BUDGET has reached its
limit========!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!');
            budget=0;
        else
            budget=1;
        end;
        accrate=acc/ite*100;
        accworsrate=accwors/wors*100;
        disp(['OVERALL acceptance rate at current temperature: ' num2str(accrate) ' %']);
```

```
    disp(['WORSE solution acceptance rate at current temperature: ' num2str(accworsrate) ' %']);

    improv=(initialOBJECTIVE-current)/initialOBJECTIVE*100;

    h=h+1;

    improvall(h)=improv;

    disp(['improvement compared to initial condition: ' num2str(improv) ' %']);

    % writing results.dat

    fprintf(fid_result, '%f\n', improv);

    fprintf(fid_result, '%s\n', '*********');

end;

save contmid;

save vmsmid;


%% writing VMS mid

cd(EightMileNet);

fid = fopen('vmsmid.dat','w');

fprintf(fid,'%0.0f\n%*10d %*10d %*10d %*10d %*10d %*10d',VMS(1,:));

for i=2:size(VMS,1)

    if numel(nonzeros(VMS(i,:)))==1

        fprintf(fid,'%7.0f\n%*10d %*10d %*10d %*10d %*10d %*10d',VMS(i,:));

    else

        fprintf(fid,'%2.0f%7.0f%7.0f%4.0f%4.0f%12.0f%12.0f\n',VMS(i,:));

    end
```

```
end

fclose(fid);

%% writing the modified control.dat mid

fid = fopen('contmid.dat','w');

cont_modified(1,1)=2;

line1 = cont_modified(1,~isnan(cont_modified(1,:)));

fprintf(fid,['%3d','\n'],line1);

cont_modified(2,2)=70;

line2 = cont_modified(2,~isnan(cont_modified(2,:)));

fprintf(fid,['%10.2f','%10.2f','\n'],line2);

fclose(fid);

%signal plan 1

fid=fopen('contmid.dat','a');

for idx=3:numberofnodes2+2

    line3 = cont_orig(idx,~isnan(cont_orig(idx,:)));  % creates the line of data without NaNs

    fprintf(fid,['%7d','%2d','%2d','%4d','\n'],line3);

end;

fclose(fid);

%

sp=0;

fid=fopen('contmid.dat','a');

for idx=numberofnodes2+3:size(cont_orig,1)
```

```
    if sum(isnan(cont_orig(idx,:)))==7 && sp==0
        fprintf(fid,'=======Two Way Stop Signs/Yield Signs Below =======\n');
        sp=1;
    end;
    line45 = cont_orig(idx,~isnan(cont_orig(idx,:)));  % creates the line of data without NaNs
    fprintf(fid,[repmat('%7d',1,length(line45)),'\n'],line45);
end;
fprintf(fid,'\n'); % a blank line between two plans
fclose(fid);

% signal plan 2
fid=fopen('contmid.dat','a');
for idx=3:numberofnodes2+2
    line3 = cont_modified(idx,~isnan(cont_modified(idx,:)));  % creates the line of data without NaNs
    fprintf(fid,['%7d','%2d','%2d','%4d','\n'],line3);
end;
fclose(fid);
%
sp=0;
fid=fopen('contmid.dat','a');
for idx=numberofnodes2+3:size(cont_modified,1)
    if sum(isnan(cont_modified(idx,:)))==7 && sp==0
```

```
        fprintf(fid,'=======Two Way Stop Signs/Yield Signs Below =======\n');

        sp=1;

    end;

    line45 = cont_modified(idx,~isnan(cont_modified(idx,:)));  % creates the line of data without NaNs

    fprintf(fid,[repmat('%7d',1,length(line45)),'\n'],line45);

end;

fprintf(fid,'=================\n');

fclose(fid);

%%



improvmargin=(current-objfun_startT)/current*100; %improvement for one temp stage

if accworsrate<5 && improvmargin<0.01

    stopcount=stopcount+1;

else

    stopcount=0;

end;

if stopcount==2

    disp('stoppiong criteria has met');

    break;

end;

T=T*0.9;
```

```
end;

fclose(fid_result);

%% writing the final modified control.dat

fid = fopen('control_final.dat','w+');

cont_modified(1,1)=2;

line1 = cont_modified(1,~isnan(cont_modified(1,:)));

fprintf(fid,['%3d','\n'],line1);

cont_modified(2,2)=70;

line2 = cont_modified(2,~isnan(cont_modified(2,:)));

fprintf(fid,['%10.2f','%10.2f','\n'],line2);

fclose(fid);

%signal plan 1

fid=fopen('control_final.dat','a');

for idx=3:numberofnodes2+2

    line3 = cont_orig(idx,~isnan(cont_orig(idx,:)));  % creates the line of data without NaNs

    fprintf(fid,['%7d','%2d','%2d','%4d','\n'],line3);

end;

fclose(fid);

%

sp=0;

fid=fopen('control_final.dat','a');

for idx=numberofnodes2+3:size(cont_orig,1)
```

```
    if sum(isnan(cont_orig(idx,:)))==7 && sp==0
        fprintf(fid,'=======Two Way Stop Signs/Yield Signs Below =======\n');
        sp=1;
    end;
    line45 = cont_orig(idx,~isnan(cont_orig(idx,:)));  % creates the line of data without NaNs
    fprintf(fid,[repmat('%7d',1,length(line45)),'\n'],line45);
end;
fclose(fid);


% signal plan 2
fid=fopen('control_final.dat','a');
for idx=3:numberofnodes2+2
    line3 = cont_modified(idx,~isnan(cont_modified(idx,:)));  % creates the line of data without NaNs
    fprintf(fid,['%7d','%2d','%2d','%4d','\n'],line3);
end;
fclose(fid);
%
sp=0;
fid=fopen('control_final.dat','a');
for idx=numberofnodes2+3:size(cont_modified,1)
    if sum(isnan(cont_modified(idx,:)))==7 && sp==0
        fprintf(fid,'=======Two Way Stop Signs/Yield Signs Below =======\n');
        sp=1;

    end;

    line45 = cont_modified(idx,~isnan(cont_modified(idx,:)));  % creates the line of data without NaNs

    fprintf(fid,[repmat('%7d',1,length(line45)),'\n'],line45);

end;

fclose(fid);
```

VMS function

```
function  VMS= vmsFUN(VMS,nodeID,move)

approachesID=find(move(:,2)==nodeID);

j=1;

for i=1:2:2*(numel(approachesID))

    vmsadd(i,1)=2;

    vmsadd(i,2)=move(approachesID(j),1);

    vmsadd(i,3)=move(approachesID(j),2);

    vmsadd(i,4)=100;

    vmsadd(i,5)=1;

    vmsadd(i,6)=70;

    vmsadd(i,7)=540;

    vmsadd(i+1,1)=move(approachesID(j),2);

    j=j+1;

end
```

```
VMS(1,1)=VMS(1,1)+numel(approachesID);

VMS=[VMS(1:size(VMS,1),:);vmsadd];
```