ABSTRACT

Title of Document: MODELING OF CONSOLIDATION BY HOUSEHOLD FOR EMERGENCY EVACUATION EVENTS

Ke Liu, Doctor of Philosophy, 2009

Directed by: Associate Professor David J. Lovell
Department of Civil and Environmental Engineering

Evacuation studies have grown in importance over the years as a number of recent emergencies, natural and man-made, have raised the general level of awareness about public responses to the threat or actual occurrence of disasters. An accurate prediction of the rates of evacuation and estimate of the time required to clear a risk area are important planning tools that can mitigate the consequences of an emergency situation.

Traditional evacuation models are predicated on the assumption that everyone would seek the quickest or shortest route to safety, given a life-threatening situation. Observations, however, show that a large percentage of the population does not seek the quickest route to safety. Parents may move toward dangers to pick up their children from

schools. Persons at work may go back home to pick up dependent family members, pets, and personal effects before evacuation begins in earnest. Incorrect assumptions of evacuee behaviors could lead to measures that negatively impact the traffic flow during evacuation.

One effective method to evaluate different evacuation strategies is the use of simulation. Most established simulation models, however, are not built to take the underlying drivers' social behavior into considerations. In this study, we develop a computerized tool for modeling evacuation dynamics with household consolidation, and then incorporate it into a traffic-simulation software platform. This tool will allow a percentage of the population to consolidate as a family before they evacuate. After that, a study is conducted to explore the consolidation by household in a network under various demand levels. A mathematical model is presented to capture the underlying relationships among the network components. Next, the traffic volumes entering and leaving the network are investigated to highlight some recommendations about the appropriate implementation of contraflow or staged evacuation strategies. To help decision makers have a better understanding of the evacuation traffic patterns, this study also examined the influences from spatio-temporal information such as the information dissemination delay, the evacuees' preparedness time, the numbers and locations of shelters in a network, and demographical information like the number of vehicles in a family.

The proposed research will allow planners to study more realistically the effects

of evacuation strategies. The results of studying such household by consolidation behavior are (1) evacuation times are significantly longer compared to the assumption of evacuees taking the shortest route away from danger in low/average demands; (2) with heavy demand, low consolidation rates can produce long evacuation times due to the rapid development of congestion at the network exits; (3) with heavy demand, high consolidation rates could delay the turning point to reverse the inbound lanes to outbound in a contraflow operation; (4) the sequencing of converting inbound lanes to outbound in a contraflow operation should start at the outermost links and work inward, due to extra bi-directional traffic on the network engaged in consolidation activities; (5) information delays and evacuees' preparedness as a family, coupled with the family consolidation behavior, are important parameters to the evacuation performance; (6) information on demographics and geography also has an important impact on the network evacuation efficiency and evacuees' social behaviors; more specifically, the evacuation performance is very sensitive to the number of shelters in the network.

MODELING OF CONSOLIDATION BY HOUSEHOLD FOR EMERGENCY EVACUATION EVENTS

By

Ke Liu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2009

Advisory Committee:
Professor: David J. Lovell, Chair
Professor: Peter C. Chang
Professor: Paul M. Schonfeld
Professor: Pamela Murray-Tuite
Professor: Richard La

Dedication

This work is dedicated to my beloved husband,

my parents, and to the memory of my grandparents.

Acknowledgement

I would like to express my sincere appreciation to my advisor, Dr. David Lovell, for his enthusiastic encouragement, continued patience, and generous support over the eight years I spent as a graduate student at the University of Maryland. This has been the most pleasant but challenging eight years in my life. This work could not have been completed without his guidance, broad and deep knowledge and patience. His passionate attitude to the scientific research and outlook on life has strong influenced me, personally and academically. Because of him, I have learned to read better, write better and even think better.

Also a special gratitude is acknowledged to Dr. Peter Chang for his helpful instructions and extensive suggestions through this study. Thanks are extended to the other committee members, Dr. Paul Schonfeld, Dr. Pamela Murray-Tuite, and Dr. Richard La for their constructive comments in this research.

I would also like to thank Dr. Hani Mahmassani, Dr. Elise Miller-Hook, Dr. Eyad Abed, Dr. Li Zhang, Dr. Wei-hua Lin, Dr. Yi-Chang Chiu, Mr. Tom Jacobs, and Mr. Phil Tarnoff, for their advice and assistance in the course of my research and studies.

Meanwhile, I would like to express my gratitude to the National Science Foundation for the partial support of this work through award number 0331707 and 0403433.

Furthermore, with a deep sense of gratitude, I especially want to thank my parents, my parents-in-law, and my grandparents, who first encouraged me to pursue the Ph.D. degree in a foreign country. I would like to express my extreme thanks to my husband, Dr. Xiang Fei, for his tremendous compassion, encouragement, and for strongly standing up for me all the way. I would not have been able to concentrate on my studies and endure some difficult times through all these years without his unconditional love.

Table of Contents

List of Tables

List of Figures

Chapter 1: INTRODUCTION

1.1 Background

An evacuation process is generally required when people are threatened by emergency events, such as hurricane, wildfire, flood, chemical and nuclear plant leak, and manmade disasters like the terrorist attacks of September 2001. While the underlying issues of evacuation decision-making are complex processes due to the uncertain nature of evacuation, an effective evacuation strategy, which has an accurate predication of the rates of evacuation and estimate of the time required to clear a risk area, is an important planning tool that can mitigate the consequences of an emergency situation, such as the chaos caused by the increase of traffic flows, and the loss of lives and properties.

Given that it is unrealistic to study and check available evacuation strategies during an actual event, numerous models and simulation studies have been conducted to investigate various problems that may be encountered during an evacuation. A review of previous studies is found in the following chapter.

Most existing evacuation modeling studies are based on the assumption that everyone would seek the quickest or shortest route to safety, given a life-threatening situation. Observations, however, show that a large percentage of the population does not seek the quickest route to safety. Parents may move toward dangers to pick up their children from schools. Persons at work may go back home to pick up dependent family members, pets,

and personal effects before evacuation begins in earnest. Incorrect assumptions of evacuee behaviors could lead to measures that, however magnanimously conceived, could negatively impact the traffic flow during evacuation.

For example, one of the evacuation strategies is to convert all traffic to outbound flow. This strategy is reasonable if evacuees do indeed exit by the shortest route. However, if consolidation as family unit takes precedence, a substantial portion of the initial traffic may be inbound. Indeed, they may travel in the direction of increased danger. Failure to model this behavior could result in chaos and gridlock during the period immediately following a disaster.

Other issues, like the evacuation information dissemination delay, the preparedness time the evacuees have for the evacuation, the demographical and geographical information like the number of vehicles in a family and the number of shelters in a network, are also interesting things to be investigated, and more specifically, the interactions between these issues and the consolidation by household behavior.

It is our premise that basic research to understand people's household consolidation behaviors during an emergent evacuation is still required. Based on our knowledge, there is no existing simulation tool that incorporates the complex social behavior that we postulate in some evacuation scenarios. Development of such a tool may not only provide more reliable output for better understanding of evacuation behaviors, but also offer more flexibility in assessing evacuation strategies through micro-simulation.

1.2 Research Objectives

The main objective of this study is to investigate the impact of evacuee's consolidated household behavior on the evacuation performance. We would also like to develop a new simulation tool that can model the underlying evacuee's social behaviors by chaining the activities corresponding to evacuation. We then simulate and evaluate evacuation strategies, such as contraflow operation and staged operation, under the impact of the consolidation by household behavior. We also model and examine how some critical factors, like information dissemination delays, evacuation preparedness, and the network demographics and geography, affect the performance of the evacuation, given the consideration of household consolidation behavior. Three sets of questions will be used to guide this research:

1. Based on the reviews of previous studies, how do we model the consolidation by household behaviors in an evacuation?

To examine this question, this dissertation first reviews extensive previous evacuation studies to identify their achievements and main limitations in modeling evacuation behaviors. To improve the existing literature, we propose a new simulation tool for modeling household consolidated behavior. The Application Programming Interface (API) is written to track multi-class vehicles' household behaviors in both typical commuting traffic and emergency evacuation. This tool can allow a percentage of the population to consolidate as a family before they evacuate. It can also model the

behaviors of multi-class drivers who are in a number of different states on the network respectively, including yet to be released vehicles, en route vehicle, vehicles that have already arrived, and vehicles in consolidation process. Furthermore, this tool can also model various chained activities, with or without scheduled previously.

2. On an existing road network, with different demand levels, how does evacuation performance vary under the influence of evacuees' household social behaviors?

In answering this question, this study develops a general network model, and then simulates different household consolidation scenarios with the tool we developed in the previous research. Different demand levels have been examined to test the interactions between the evacuation behaviors and the traffic conditions. A mathematical model is also presented to capture the underlying relationships among the network components.

3. In the evacuation, how might different strategies, such as contraflow and staged evacuation, be implemented properly with household consolidated behaviors being considered?

Based on previous research finding from the household consolidation behavior, this study will seek to investigate when to implement lane reverse strategy and how. As discussed earlier, since people incline to consolidate as families and then evacuate as a single unit, when and how to change traffic patterns from inbound to outbound are critical. Therefore, this study will model and investigate how the traffic flows entering

and leaving the network, in the incorporation of people's household consolidation behavior.

4. In the evacuation, how might other critical factors, such as information dissemination delay and network demographics and geography, affect the evacuation performance in the incorporation of the household consolidation behaviors?

To extend the study of the evacuation dynamics with household consolidation, this dissertation also studies the impact from spatio-temporal information and demographics information. The ability to observe flow patterns and performance characteristics in evacuation in the incorporation of both evacuees' household consolidation behaviors and the spatio-temporal information presents a challenge for transportation agencies. In this study, the interaction between the household consolidation behavior and spatio-temporal issues, such as the information dissemination delay, the evacuees' preparedness time, the numbers of shelters in a network, and demographics information like the numbers of vehicles in a family, have also been investigated.

1.3 Organization of the Dissertation

The dissertation is divided into eight chapters. Chapter 1 describes the general background of this study, and introduces the research objectives guided by a set of questions. Chapter 2 provides a general overview of previous evacuation models and related literature. Chapter 3 presents the development of a simulation tool that describes the efforts that have already been undertaken as part of this study to model the household consolidated behaviors in an emergent evacuation. Chapter 4 explores the consolidation by household in a network under various demand levels. Chapter 5 investigates the traffic volumes entering and leaving the network, to reveal the impact of family consolidation to the development of contraflow and staged evacuation strategy. Chapter 6 discusses the evacuation dynamics with household consolidation under extended considerations, which include the evacuation information dissemination delay and the preparedness time the evacuees have for the evacuation. Chapter 7 examines how demographics and geography information, like the number of vehicles in a family, and the number of shelters in a network, might have an impact on the evacuation performance in the incorporation with the consolidation by household behaviors. Finally, some conclusion and possible future research directions of this study are presented in Chapter 8.

Chapter 2: LITERATURE REVIEW

The background relevant to this study is divided into four sections. In the first section, an overview of evacuation literature is presented. The second section looks into the existing evacuation simulation models. The third section investigates observations to evacuation behavior patterns. Finally, a summary of the literature review is offered.

2.1 Research Overview

With the terrorist attacks of September 2001 and the Hurricanes Katrina and Rita in 2005, researches for evacuation-related transportation issues have received more and more attentions in recent years. This section outlines the relevant research in emergency evacuation, for example, the application of contra-flow operations and staged or phased operations, the integration with Geographical Information System (GIS) data, the implementation of Intelligent Transportation System (ITS) technologies, the refinement of urban signal controls, the involvement of mass transit, and so on. The detailed review of the literature gives us a clear idea about how effectiveness of such studies is and what we can learn from them.

To begin with, it should be pointed out that emergency evacuation operations are always divided into different phases (Southworth 1991, Sisiopiku et al. 2004, Murray-Tuite 2003). Those phases generally consist of preparedness, response and recovery processes.

Most studies we discussed here are focused on the preparedness and response phases, which are associated with the actual evacuation.

Emergency preparedness typically involves the development of detailed emergency plans that address the roles, responsibilities, and actions required by local and state emergency agencies. In development of such plans, Geographical Information System (GIS) plays a key role by providing useful geometrical data and demographical information.

Early in 1993, Pidd et al (1993) proposed a CEMPS simulation model which used GIS system as the data base, and then linked the GIS data to a developed micro-simulator. The CEMPS microscopic simulator was developed using an object-oriented version of the three-phase approach to discrete simulation. Pidd's model does not take traffic congestion into consideration. The CEMPS simulator cannot simulate the traffic control in the road network either. Later, in 2000, Silva and Eglese (2000) enhanced the CEMPS simulation model's decision support capabilities. In this enhanced version, the model is able to communicate to a GIS-ARC/INFO database, which can perform mapping, plotting, storing, handling and analyzing spatial data. Relevant studies include Dunn and Newton (1992), Pal et al. (2003), Wilmont and Meduri (2005), and Laefer et al. (2006). However, common character is that all the proposed approaches/systems are developed for contingency planning in evacuation rather than for real-time emergency management use.

Many other factors may also affect the development of emergency plans. For example, low-mobility people such as those in schools, nursing homes and hospitals require public transportation or walk as pedestrians to go to shelters. El-Mitiny et al. (2007) studied the effect of evacuating persons either by bus or as pedestrians in a small road section of Orlando, Florida. By using VISSIM to simulate nine scenarios as percentage changes in volume of pedestrian and number of bus on the evacuation network, results of this study suggested that giving signal priority for buses with consideration of pedestrian movements during evacuation would reduce evacuation time. However, further discussions of how to improve or design transit plans in evacuation were not given.

With the development of Intelligent Transportation System (ITS) technologies, there are more and more studies involved ITS technologies into emergency evacuation planning. For example, Morrow (2002) proposed how to use Dynamic Message Signs (DMS) to inform and assist evacuees in Orlando, Florida for a safe, efficient evacuation. Lively et al. (2006) discuss how Advanced Traveler Information Systems (ATIS/511) enhance emergency and disaster response. The development of ITS technologies into emergency evacuation planning helps transfer the proposed evacuation strategy into the field practice.

Another large number of literatures are associated with evacuation response phase. In this phase, people response to the emergency by following the order of evacuation. A large number of evacuation strategies are studied to provide evacuees better route-guidance and effective movements.

For example, one way to facilitate evacuation is appropriate traffic control. Sisiopiku et al (2004) studied the effect of evacuating a particular network by using CORSIM to simulate various evacuation plans. The results suggested that signal optimization for the evacuating traffic could decrease average vehicle delay and increase total evacuation time.

Chen and Miller-Hooks (2007) constructed a simulation model via CORSIM and tested various signal timing plans for Washington D.C. area in no-advance-notice disasters. Results of this study revealed that increasing signal cycle length for both evacuation route and minor roadways would provide the best result in most full-scale evacuations.

At the same time, Liu (2007) also studied the signal control strategies for Washington D.C. area under emergent evacuations. This study also employed CORSIM simulation as a base model to simulate different signal control plans and then employed a mathematical model to generate refined signal timing plans quantitatively. Results from this research indicated that critical intersections play important roles and demand distribution could significantly influence the effects of different control strategies.

Although the lack of route choice capability of CORSIM makes such studies extremely difficult, the results from these researches do reveal that simulation is a good option for developing and evaluating evacuation plans. An in-depth overview of simulation models in evacuation study will be presented later in this chapter.

Staged evacuation, also known as phased or zoned evacuation, is another commonly used strategy in evacuation response. In this control strategy, evacuees are evacuated by zones in a particular sequence.

For example, Chien and Korikanthimath (2007) proposed an analytical model to optimize the number of evacuation staged zones for minimum evacuation time and delay time. Chen and Zhan (2008) investigated the effectiveness of simultaneous and staged evacuation strategies in different road networks using agent-based simulation. Mitchell and Radwan (2006) identified critical factors such as population density, distance to destination, flow rate and so on which might affect the staging decisions. Related studies consist of MWCOG (2004), Farrell (2005), Liu et al. (2006), etc.

Although this type of studies identified the effectiveness of staged or phased strategies in emergency evacuation, social behavior is not considered. Evacuation strategy and tactics can potentially change the evacuation pattern significantly.

One evacuation strategy that has received significant attention in the literature is contraflow. Under contraflow operations, one or more of the inbound lanes are used for outbound evacuation. Contraflow evacuation has been shown to be a useful method in increasing the flow rates of evacuation traffic largely (Wolshon 2001, 2002).

Contraflow strategy study is mostly handled by scenario analyses and simulation models. For example, Theodoulou and Wolshon (2004) studied the designs of contraflow at the entry points and evaluated the contraflow plans at the city of New Orleans. They coded contraflow operations into CORSIM simulation platform by adding barricades between lanes and making closure of normal flow lanes to represent the planned contraflow operations. Results from this study indicated that the use of two contraflow lanes could increase the capacity of a four-lane freeway by about 53 percent. Their results also revealed that inappropriate design of entry points may create new bottlenecks which further lead to heavily congested zones during an emergent evacuation.

Lim and Wolshon (2005) focused on the designs of contraflow at termination points, where traffic flow changed from reverse flow patterns back to normal flow directions. The study also employed CORSIM to model the planned operations by creating a permanent blockage incident for lane closures. The results from simulation suggested that factors, such as split design, merge design, channelization and separation design, would help enhance the traffic flow performance through the termination vicinity.

Kwon and Pitt (2005) studied the access capacity for contraflow strategy design by using the simulation software DYNASMART-P to test alternative plans for evacuating downtown Minneapolis. To code the contraflow operation, an incident was crated to reduce half of capacity of outbound links until the evacuation started. When the contraflow operation starts, the incident was moved from outbound links to inbound links to block these inbound links completely. Kwon and Pitt showed that the access capacity

to the network is the critical issue in contraflow operations. For example, when the capacities of the key entrance ramps in the Minneapolis downtown area were increased, the performance of the contraflow operations was also improved.

Similar studies include Zou et al. (2005), Wolshon and Lambert (2005), and Williams et al. (2007). Although the large number of contraflow studies proved the effectiveness of contraflow operations in improving evacuation efficiency, it should be noted that these studies do not take social behaviors into consideration. For example, if evacuees consolidate as families before evacuation in earliest begins, changing the traffic pattern from inbound to outbound could exacerbate the problem. To our knowledge, there is no work to investigate this variation.

Recently the analytic approaches have also been proposed to study contraflow operations. For example, Tudyes and Ziliaskopoulos (2004) proposed a link-coupling approach using the cell transmission model for deciding which reverse lanes should be utilized under evacuation. With solving the objective function of the system travel time optimization, they generated a contraflow plan. Their results indicated that the proposed contraflow plan had a significant reduction in total travel time comparing with a normal evacuation plan without any lane reversibility.

In the extended work, Tudyes and Ziliaskopoulos (2006) proposed a Tabu-based heuristic approach for designing an optimal contraflow plan for urban evacuations. Their objective is to find the optimal capacity reversibility from a given capacity re-distribution

reversibility scheme. Their results revealed that even though the solutions for the large-scale capacity reversibility problem are most likely sub-optimal, this approach would still provide some quantifiable useful contraflow designs.

Similar analytic approaches were presented by Shashi and Kim (2006), and Liu (2007). Although these analytic approaches can generate contraflow plans with various optimization formulations, unlike simulation models, these studies lack the abilities in effectively modeling traffic behaviors and capturing network traffic dynamics.

To sum up, from the review of previous studies, we found that simulation models were widely applied in different types of evacuation studies. Results from the literature shows that simulation is a good option for developing and evaluating evacuation strategies. Therefore, in the following section, we will review and discuss the available simulation models in representing and evaluating emergency evacuation process.

2.2 Traffic Simulation Models in Emergency Evacuation

A large number of different models, especially simulation models, have been developed to represent traffic conditions under emergent conditions and to provide route-choice guidance to evacuees.

Sheriff and Mahmassani (1982) developed a fixed-time NETwork emergency eVACuation modeL (NETVACL) which included a route-choice model for estimating network clearance time in the context of nuclear emergencies. Hobeika et al. (1985) proposed a MASS eVACuation model MASSVAC which used macroscopic simulation to estimate the maximum network evacuation times. However, due to the relatively inadequate computer technology at that time, these models are primarily static analysis tools at macroscopic or mesoscopic levels. Such models do not attempt to capture traffic dynamics, nor does it track detailed movements of individual vehicles.

Later on, Oak Ridge National Laboratory developed a microcomputer-based modeling system OREMS to simulate traffic flow during regional population evacuations (Rathis and Solanki 1993). However, they assumed that en-route travel time is the only factor in distributing traffic. That is to say, more traffic will go to the closet destination, which may not be realistic during evacuation.

Pidd et al (1993) proposed a microscopic simulation model called CEMPS, which used GIS system as the data base. The CEMPS microscopic simulator was developed using an object-oriented version of the three-phase approach to discrete simulation. Silva and

Eglese (2000) enhanced the CEMPS simulation model's decision support capabilities by linking the model to a GIS-ARC/INFO database, which can perform mapping, plotting, storing, handling and analyzing spatial data. However, the CEMPS simulation model assumes that the vehicles' traversing time is independent of the network congestion. The CEMPS simulator does not simulate the traffic control in the road network.

Based upon the MASSVAC, Hobeika and Kim (1998) proposed a transportation evacuation decision support system (TEDSS) model to incorporate new modeling features such as the User Equilibrium assignment algorithm. They use this model to analyze and develop evacuation plans and management for different scenarios. However, this study also assumed that all vehicles in the network would travel towards the closest exit point outside the dangerous area without traveling on the links leading towards the center point of the area. The possibility of an evacuee entering the danger area to retrieve belongings or people is not considered.

Rather than those efforts in developing simulation models for evacuation, more researches investigated evacuation problems from different perspectives using existing simulation software packages. Among them, the most widely applied simulation software packages are TSIS/CORSIM, Paramics, VISSIM, and DTA-based simulation software.

Traffic Software Integrated System (TSIS) is a suite of traffic analysis tools including its core simulation model CORridor SIMulation (CORSIM), developed by U.S. Federal

Highway Administration since 1970's (ITT Industries, 2000). In evacuation studies, TSIS/CORSIM has been widely used to develop emergency evacuation planning systems (Zou et al. 2005, Liu et al. 2005), evaluate contraflow strategies (Lim and Wolshon 2005, Liu 2007), staged or phased strategies (Liu 2007), signal operations (Sisiopiku et al. 2004, Chen et al. 2007, Liu 2007), geometric design alternatives (Theodoulou and Wolshon, 2004), the effect of traffic incidents and events (Chen et al., 2007), and so on.

PARAller MICroscopic Traffic Simulator (PARAMICS) has been developed by Quadstone Ltd., Edinburgh, Scotland (Quadstone 2006). It provides not only traffic simulation with 3-D visualization but also a powerful programmer module which allows users to augment the core Paramics simulation with new functions. Church and Sexton (2002) employed Paramics to test the efficiency of a proposed bulk lane demand model for wildfire evacuation planning in Santa Barbara, CA. Here the bulk lane demand meant the total vehicle demand leaving a neighborhood versus the number of lanes of roadway leaving a neighborhood. The studied road network was coded into Paramics from a database provided by the Geology Department at UCSB. Eight different scenarios with various traffic controls and road exits were examined. Results from simulation revealed that without special evacuation plans, the neighborhood with high bulk lane demand may not be able to evacuate in a timely manner during a wildfire. This study considered people's taking care of last minute issues before departing, such as take their pets or gather a few belongings, however, rather than modeling this kind of behaviors, it simply represented these behaviors by assuming that 30% total demand leaves in the first 5

minutes, 50% leaves within the nearest 5 minutes, and 20% leaves within the next five minutes.

Cova and Johnson (2002) used Paramics to study neighborhood evacuation planning in Salt Lake City, UT. In this study, statistical data collections were based on individual households within a neighborhood. A US Geological Survey digital orthophoto quad (DOQ) was acquired to code the transportation network and household structures into Paramics. This study investigated a set of neighborhood evacuation scenarios, which was comprised of a combination of two variables, i.e. number of evacuating vehicles per household and the mean vehicle departure time. For example, a given scenario might be one in which the average number of vehicles per household was 0.5 (few residents at home) at the time of the event and the average vehicle departure time was 5 minutes (evacuees have quick response). An API tool was presented to manage vehicle generation, departure timing and destination choice for the studied network. Results from simulating those scenarios indicated that shorter household preparation times would result in a quicker evacuation for the study area. However, very urgent evacuations would cause significant traffic congestions. Therefore, construction of a second access road and staged operations were suggested. However, it should be noted that, although this study focused on a neighborhood evacuation on a household basis, it assumed all vehicles were assigned from their home to their closest exit or shelter by using the network distance.

Cova and Johnson (2003) proposed a mixed-integer programming model to study lane-based evacuation routing plans for downtown Salt Lake City. Their objective is to route vehicles to their closet evacuation zone exit, as well as minimizing the number of intersection merging-conflicts and avoiding intersection crossing-conflicts. To evaluate the relative efficiency of the optimal lane-based routing plans comparing with the no-routing plan, Paramics was used to run the experiments. Results revealed that channeling flows at intersections to remove crossing-conflicts could significantly decrease network clearing time over no routing plan. It also revealed that the benefits of channeling flows to remove merging could as well reduce network clearing time, though this amount of reduction was likely to vary depending on the road network context and scenarios. However, it should be pointed out that, due to evacuee's social behavior, the objective of this study – to route vehicles to their closet evacuation zone exit – may not represent the reality.

VISSIM has been developed by KLD Associates, Inc. in German (PTV, 2005). It is one of the latest developed microscopic traffic simulation packages, featured for multi-modal traffic flow modeling, such as pedestrians, cyclists, and motorized vehicles. El-Mitiny et al. (2007) employed VISSIM to study the effects of transit transportation and pedestrians on emergency evacuation planning, as discussed earlier in previous section. Other works include evaluating lane reversal plans (Tagliaferri 2005, Williams et al. 2007), investigating traffic operations (Han and Yuan, 2005), etc.

In order to better estimate and predict traffic conditions with respect to changing demands, researchers have attempted to develop traffic simulation models that implement dynamic traffic assignment (DTA) algorithms. Among them, DYNASMART-P, developed by FHWA in the 1980s, has been widely applied into evacuation study. DYNASMART-P is a meso-scopic model that has been used to study contraflow evacuation operations (Kwon and Pitt, 2005), develop evacuation routing plans (Chiu et al. 2005), investigate staged or phased evacuation strategies (Sbayti and Mahmassani, 2006), and analyze evacuees' trip-chain decision making activities (Murray-Tuite and Mahmassani, 2003, 2004). Due to the similar premise as our research, Murray-Tuite and Mahmassani's studies will be discussed in more details in the Section 2.4.

DYnamic Network Assignment for the Management of Information of Travelers (DYNAMIT) is another DTA program developed by FHWA in the 1980s. Balakrishna et al (2008) applied DYNAMIT for the modeling of transportation network performance under emergency conditions. They tested the proposed modeling framework to the City of Boston using a contraflow strategy implemented. Their results showed that the provision of reversed traffic flows could reduce the average travel time. However, this study did not take evacuee's social behaviors into consideration.

In summary, from the numerous simulation-based evacuation studies mentioned above, the findings reveal that effectiveness of applying transportation simulation packages into evacuation studies. However, there is no conclusion that one model or package is superior to the others. Each model or package has its own strengths and weaknesses.

None of them can be used for all situations. It all depends on the particular applications when pick up the appropriate model for research.

2.3 Evacuation Behavior Patterns

In order to model evacuation appropriately, choices and behaviors of evacuees must be considered. According to social science research, there are phenomena, distinctive to evacuations, which differ from the ordinary traffic situations.

Many of previous social science studies have reported that household evacuations are an important form in evacuation. Zeigler et al. (1981) point out that the majority of evacuees from Three Mile Island departed as complete family units, yet there were a larger number of partial family units departing than previous research on natural disasters would have suggested. Urbanik (2000) correctly identifies "returning commuters" as a population segment that might tend to consolidate by household before evacuating. Wolshon (2001) points out that evacuees frequently travel with pets, children, and elderly family members. Other studies, such as Greene et al. (1981), also point out that people tend to evacuate as a family unit or as an established social unit such as a carpool. Similar studies include Barrett et al. (2000), Alsnih et al. (2005), and so on.

Disaster researchers have also found that almost all families evacuate using private vehicles (Drabek, 1986; Tierney et al., 2001). In part, this is because personal vehicles are mobile assets. This implies that those who commuted by car will attempt to leave by car. Most of those who used other modes to commute to work will attempt to evacuate from work using the same mode, but situational circumstances might force them to improvise by taking another mode.

Almost all research on disasters has studied families' evacuation from their homes, and where the most frequent destinations are the homes of relatives and friends who live in what is believed to be a safe location (Drabek, 1986; Tierney et al., 2001). This implies that the initial evacuation destination from the workplace will be the home unless the family has established a plan to meet elsewhere. Once the family is reunited, or has at least established that all members are in safe locations, they will travel to a location of temporary shelter if they believe that they need to evacuate from their homes.

People will attempt to evacuate on the most familiar routes unless circumstances dictate otherwise (Dow & Cutter, 2002; Prater et al., 2000). The initial plan will be to evacuate from work to home via their usual commuter route, but they will use alternate routes if their preferred route is unavailable.

Research has repeatedly found that panic is hardly observed during major evacuations (Herr, 1984). It is also observed that there is no state of panic during toxic chemical releases – which would be the situation most likely to produce panic. A few studies (Sattayhatewa et al., 2000, Zelinsky 1991) reported that people only panic during evacuation in some special environments, such as limited visibility, limited exits. Therefore, generally, we can model evacuation with a traffic simulation model without driver panic.

Recently researchers also studied evacuees' trip-chain behavior. Murray-Tuite and Mahmassani (2003, 2004) studied behavioral aspects of family trip-chain evacuation

employing DYNASMART-P. In their work, an initial set of link travel time was generated by DYNASMART-P from planning/historical data for a certain network. With these perceived travel time, each household's decision maker could determine where to meet and how to meet by solving two linear integer programming models. For example, if this family has school children, how parents pick up their children on their way to family's meeting place. After solving the linear models, the results of each family's trip-chain sequencing decision were input into the DYNASMART-P model. The simulation model then evaluated the effects of various network loading strategies. Their results revealed that a minimum of 150% of the original demand should be assumed for developing evacuation plans if the impacts of trip chains are ignored.

Though this study demonstrated a reasonable way to investigate the evacuation trip-chain behaviors, it has some limitations. For example, it used the planning/historical O/D data to estimate each family's travel time; and it predetermined the waiting time for multiple vehicles in one family. A detailed discussion about this study and our research will be reported in chapter 4 in this dissertation.

2.4 Summary of Literature Review

Extensive research has been conducted to study emergency evacuation preparedness and response processes. However, as reviewed in the previous sections, current simulation models and evacuation studies have some important limitations, which make those models inconsistent with evacuees' social behaviors under emergent evacuation situations. The followings summarize the problems and limitations of current evacuation studies.

- The overview of evacuation literature covers the utilization of Geographical Information System data, the integration of mass transit, the implementation of Intelligent Transportation System technologies, the refinement of urban signal controls, and the development of contraflow operations and staged operations. However, in these studies, the understanding of the impact of evacuees' social behavior on traffic flow during emergency evacuation is limited.

- Simulation models have been widely developed and applied into different types of evacuation studies. Results from the literature show that simulation is a good option for developing and evaluating evacuation strategies. However, there is no guideline for choosing appropriate simulation models for a certain evacuation study.

- Rather than those efforts in developing simulation models for evacuation, more researches investigated evacuation problems from different perspectives using

existing simulation software packages or models directly. As a result, limitations and shortcoming in the kinds of software themselves could also be involved, which make them unreliable to work as platforms in solving particular problems.

- Evacuation modeling studies in the research literature demonstrate various approaches to represent traffic conditions under emergent evacuation conditions. Some of them have concluded that household consolidation is an important issue for certain types of evacuations. However, the specific social behaviors of household consolidation have not yet been fully explored.

As a product of this study, we are going to propose a new approach for better understanding evacuees' social behaviors and special traffic flow patterns during evacuation. To do this, we will develop a computerized tool that has high flexibility and reliability to model various types of household consolidation behavior in a mass emergency evacuation event, which might overcome and improve these limitations and shortcomings of the existing evacuation simulation model. This tool can model the behaviors of multi-class drivers who are in a number of different states on the network. We will also demonstrate how to use our tool to assist traffic engineers for doing evacuation planning and operations, such as contraflow strategies, in a better way.

Chapter 3: Development of API tool

This chapter presents an approach to develop a simulation-based tool to study emergency evacuation that includes household evacuation behaviors. The Application Programming Interface (API) is written to track household behaviors of vehicles with various dispositions in both typical commuting traffic and emergency evacuation.

This chapter is organized as follows. The first section presents the modeling framework (corresponding to the first objective identified in section 1.1. of chapter 1). The second part of this chapter introduces the hierarchical data structure. In the third and forth section, the initialization of the data structure and the sorting of the data structure are presented. The fifth section of this chapter introduces how the API tool deals with the vehicles in typical commuting traffic. Next, section 3.6 to 3.9 present four designed modules, which study the behaviors of drivers with various dispositions who are in a number of different states on the network. These states include yet to be released vehicles, en route vehicle, vehicles that have already arrived, and vehicles in the consolidation process.

3.1 System Framework of API Tool

We present a framework that represents the development of the Application Programming Interface (API) tool as illustrated in figure 3-1. It consists of seven principal modules that provide the abilities to do different studies. The main function of each module is briefly stated as follows.

The **household generation** module is designed to initialize and sorting a hierarchical data structure. The hierarchical data structure consists of a family structure and a vehicle structure, which has interrelations between each other. In this way, each family's information in the network is available, and vehicles within a family can be tracked during the simulation. A detailed presentation of this module, including the design, initialization, and sorting of the data structure, is in the section 3.2, 3.3, and 3.4 respectively.

At the beginning of the simulation, all vehicles that are loaded into the network are assigned origins and destinations, a release time, and its associated home base. The **regular simulation** module is designed to model driver behaviors for the vehicles in typical commuting traffic. We are doing the commute traffic simulation as part of the initialization to load the network. A detailed discussion of this module is in the section 3.5.

Then an emergency event is initiated at a predetermined time. At this time, the vehicles that have not yet been released into the network will be assigned a new destination and a

new release time according to their states in the network. For example, for a vehicle that is still at its work and plans to go home a few hours later, this vehicle will be released immediately. The module of **yet to be released vehicles** is designed to model driver behaviors for those vehicles that have not yet been released at the onset of the evacuation. A detailed description of this module is in the section 3.6.

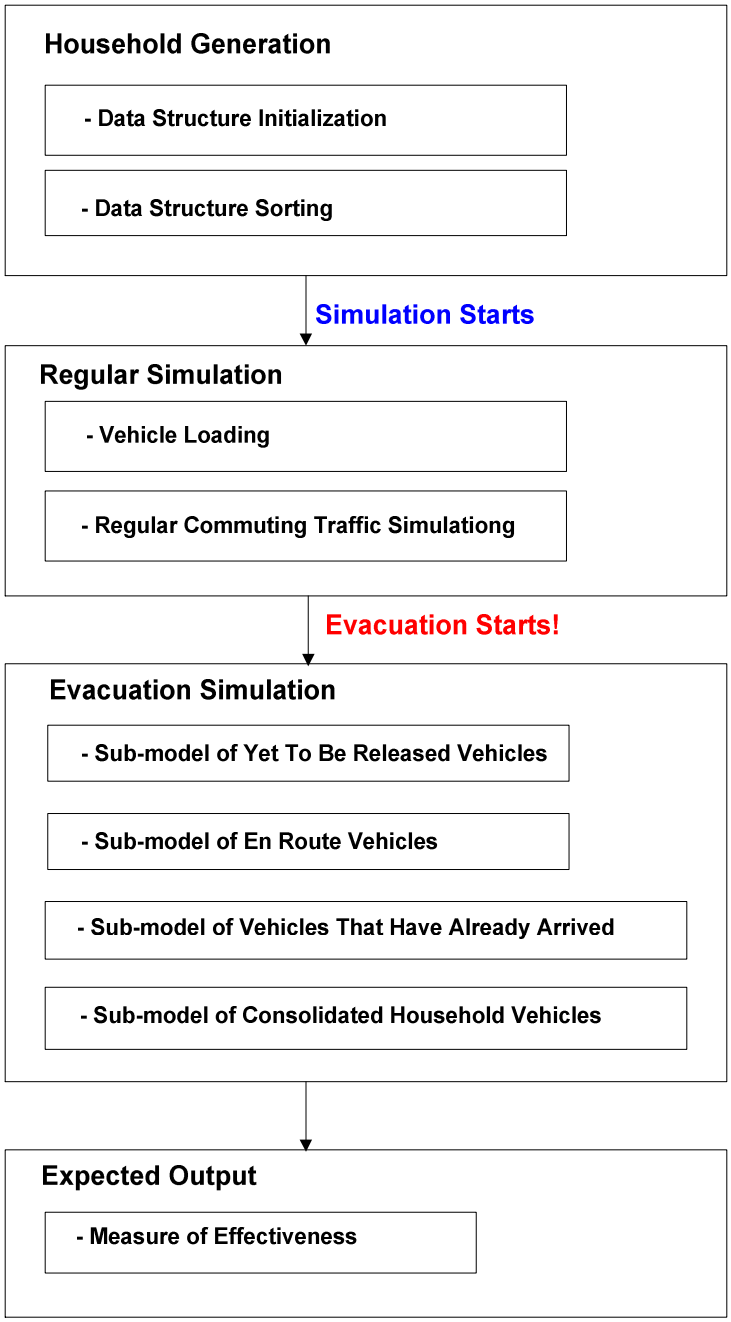


Figure 3-1. System framework

At the beginning of the emergency event, some vehicles may realize that there is an evacuation while they are still en route. Each of these vehicles will be assigned a revised destination that will only be enforced when a random revision time has passed, and in the

meantime they will continue on their original routes. When the revision time is reached, this kind of vehicles will travel towards the new destination dynamically. The module of **en route vehicles** is designed to study driver behaviors for those vehicles who have to change their destinations en route due to the evacuation. A detailed discussion of this module is in the section 3.7.

Finally, when a vehicle arrives at its destination, for those who are not affected by the evacuation (i.e. they arrived at their destinations before the onset of the evacuation), a new trip will be generated for each of them based on their trip chain activities. For those vehicles who are affected by the evacuation, we will check if each of them is a consolidated vehicle. If so, consolidation actions are invoked. Otherwise, the vehicle will be sent to the closest shelter/exit. The module of **arrived vehicles** is designed to study the dynamic behaviors for those vehicles who just arrived at their destinations. A detailed presentation is in the section 3.8.

One of the key features of this developed API tool is to model the effect of household consolidation on evacuation traffic. There are two types of vehicles: consolidating vehicle and non-consolidating vehicle. As we mentioned before, once the evacuation starts, a non-household vehicle will evacuate to a shelter immediately, while a household vehicle will return to its consolidated point. After all vehicles associated with a household arrive at their home, they then evacuate together in a single vehicle, i.e, a consolidated trip is generated. Therefore, the module of **consolidating household**

vehicles is designed to study the effect of this vehicle's household consolidation process on the traffic. A detailed description is in the section 3.9.

At last, the **output** module is designed to collect data that shows how the consideration of household consolidation behaviors would affect the network performance under evacuation situation. The evacuation metrics are the percentage of arrival vehicles, the number of evacuated vehicles, the vehicle miles traveled and the average travel time. It will be used in the case study of the Chapter 4.

3.2 Introduction of Data Structure

Assuming that evacuees seek the shortest or quickest route to safety is an over-simplification of the evacuation problem. Depending on the nature of the emergency evacuation, a significant portion of the evacuees is expected to consolidate as a family unit before evacuation away from the area of danger, especially in cases of planned evacuation, such as hurricane. Past researches have attempted to model this behavior by assigning a portion of the vehicles to turn into the danger zone (Sisiopiku et al. 2004, Chen and Miller-Hooks 2007, El-Mitini et al. 2007). The accuracy of such a model depends on how the number of turns into the danger zone is determined. A more accurate approach should consider a few criteria, such as how many family members drive to work, how many members need to be picked up, the current location of the vehicles, where the members of the evacuation unit should consolidate, whether all members of the evacuation unit arrived at the consolidation point or not, how much delay is there before evacuation should begin, and for those family members that finish their planned trips, does s/he have any trip chain activities. .

One way to keep track of such information is to assign a data structure to each family, and a data structure to each vehicle. If a data structure containing such information can be added to a general purpose traffic simulation tool, then it may be possible to use the general purpose tool to simulate evacuation traffic. As illustrated in table 3.1, the following data would be necessary for such a simulation:

Table 3-1. Summary of Data Information

Data Name	Data Description	Data Utilization
HomeLocation	Home location (integer) for each family	Generate trip OD for family members
WorkLocation	Work location (integer) for each vehicle	Generate trip OD for family members
NumOfVehs	Number of vehicles (integer) in a family's fleet	Count number of trips completed & trigger consolidated evacuation trip
NumOfArrivedVehs	Number of vehicles (integer) arrived their associated family (consolidation point)	Decide when consolidated evacuation trip starts
Homebound	Type of Homebound (Boolean) (i.e. family-dependent)	Decide whether the family's fleet are consolidated household vehicles
orgZone	Origin zone (integer) for each vehicle	Generate trip OD for vehicles
desZone	Destination zone (integer) for each vehicle	Generate trip OD for vehicles
RevisedDest	A revised destination (integer) for an en route vehicle that is aware of the evacuation	Enforce the vehicle to travel to a new destination when a random revision time has passed
StartTime	Departure time (float) for vehicles leaving their origins	Decide when vehicles will be released
RevisionTime	Revision time (float) for an en route vehicle that is enforced to change its destination due to the evacuation	Decide when the en route vehicle will travel towards its revised destination
AwareTime	Aware time (float) for vehicle being aware of the evacuation	Decide at what time the vehicle (yet to be released or en route) is aware of the evacuation
DelayTime	Delay time (float) for vehicles who are waiting to start a new trip	Generate a random waiting time in a specific range for family members

We propose to set up two data structures for this purpose: (1) a structure containing information about each family, and (2) a structure containing information about each vehicle of the family. The structure contains the following information:

```
Structure Family {
        int NumOfVehs;
        int NumOfArrivedVehs;
        int HomeLocation;
        int HomeBound;
        Struct Vehicle*Vehicle_list;
        Struct Family*next;
};

Structure Vehicle {
        int WorkLocation;
        int HomeLocation;
        int orgZone;
        int desZone;
        int RevisedDest;
        Float StartTime;
        Float RevisionTime;
        Float AwareTime;
        Float DelayTime;
        Struct Vehicle*next;
        Struct Family*next_family;
};
```

The proposed data structure has two significant features: i) the application of linked list; and ii) its hierarchical constitution.

Instead of using conventional data structures such as an array to store family and vehicle data, we employ linked list to store the information. The advantage of using a linked list is the ability and flexibility to add or remove a node that contains family or vehicle information as shown in the figure 3-2 into or from the linked list at any time and any location, while keeping the link list connectivity. In this study, this feature is extremely important as consolidated vehicle are represented by a new vehicle. The ability to insert such a new vehicle without changing the other vehicles release time into the network is an important feature.

For example, there is a vehicle that arrives at its work location at 9 a.m., and will leave for its home at 5 p.m. In most of micro-scopic simulation tool like Paramics, once a vehicle arrives at its destination, it disappears. Therefore, trip chaining is represented by a new vehicle. This new vehicle's destination is the home of vehicle being replaced and its release time is set to 5 PM. The new vehicle is then inserted into the sorted vehicle linked list among the yet to be released vehicles. If we model this kind of vehicle maneuver with the traditional arrays, we have to determine the size of the vehicle array upon its creation. Since we do not know the number of this kind of vehicles without running the simulation, it is difficult to decide the right size of the array. If the per-determined array size is too small, we lose the capability to generate enough new vehicles. If too big, the memory and computation efficiency is lost.

Moreover, since the family linked list and vehicle linked list are connected, a hierarchical linked-list structure is designed to allow inter-correlations between each family and its associated members. In this way, for a certain family, we can track each family member's information like their destinations, release times, etc. For a certain vehicle, we can also find which family it belongs to, how many members this family has, and so on. In this work, this feature is particular important to model drivers' consolidated household evacuation behavior. For example, once a vehicle arrives at its consolidated point, we need to check if all vehicles associated with the same household have already arrived at their home. If so, a new consolidated trip will be generated. By using this hierarchical data structure, such information can be exchanged between the family class and the

vehicle class at any time during the simulation.

Graphical representations of these linked lists are shown in Figures 3-2 below.

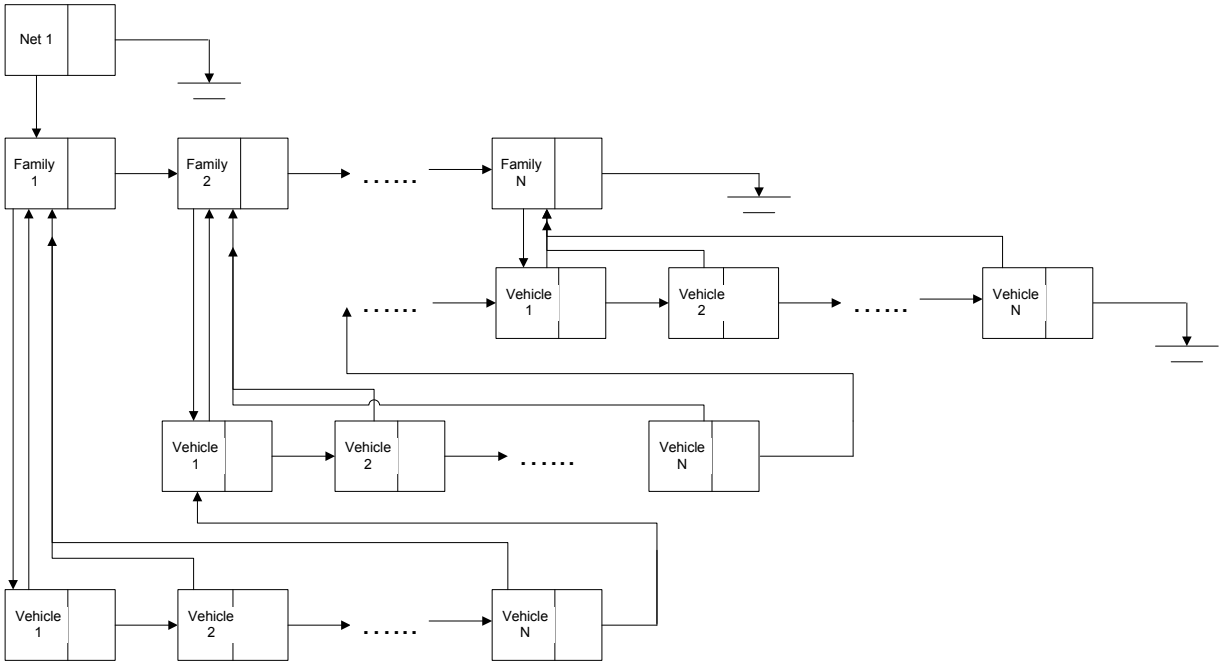


Figure 3-2. Family and vehicle linked list structures for household evacuation simulation

3.3 Data Initialization

To initialize the hierarchical data structure constructed in the last section, we need to fill in information for each family in the network, and for each vehicle of the family. Such information includes: (1) the distribution of family locations, (2) the distribution of the family member(s)' work locations, (3) number of vehicles in each family, (4) is this family a consolidated one, (5) the distribution of departure time of the vehicles, and (6) the distribution of origins and planned destinations of the vehicles.

Paramics API functions do not have the ability to generate those family-related and vehicle-related data to fulfill the aforementioned requirements. Therefore, a couple of random number generators are created to generate the random variates based on a priori assumed distributions. Rationale and implementations for each of these is explained further in the following paragraphs.

First, we assume that the distributions of the family locations and work locations are bi-normal distribution. This is an attempt to represent a situation where the city center is the central business district (CBD), which has a concentration of commercial facilities, and where there are also some residential zones around the CBD. The farther away from the city center, the fewer families and commercial facilities exist. A pair of normal random variates is generated for this purpose. Polar Method (Law and Kelton, 2000) is used as follows.

Step 1. Generate a pair of uniform random variates which are independent and identically distributed with U(0,1), say, r_1, r_2

Step 2. Let $v_1 = 2r_1 - 1,\ v_2 = 2r_2 - 1$, $w = v_1^2 + v_2^2$

Step 3. If $w > 1$, reject, go to step 1. Otherwise, let $y = \sqrt{\frac{(-2\ln w)}{w}}$

Step 4. Let $x_1 = v_1 y, x_2 = v_2 y$, then x_1 and x_2 are independent and identically distributed normal random variates ~ N(0, 1).

Second, as the size of the studied network varies, the generated normal random variates need to be scaled and then discretized into the network by adjusting the standard deviations of the variates. In the context of this research, this refers to the centralization of locations of work zones and home zones. Since some of the zones are on E-W locations while others are on N-S locations, a Bernoulli random number is generated to decide which direction each zone locates. The Inverse Transform approach (Ross, 2006) is utilized as follows. Besides, the function *LocationGenerator()* is coded to handle the setting of home and work locations as just described.

Step 1. The distribution function is:

$$F(x) = \begin{cases} 0 & if\ x < 0 \\ 1 - p & if\ 0 \le x < 1 \\ 1 & if\ 1 \le x \end{cases} \qquad (2)$$

Step 2. Generate $U \sim U(0,1)$

Step 3. If $u<(1-p)$, return $X=0$. Otherwise, return $X=1$.

Third, in this case study, we adopt the following simple distribution of vehicle ownership: 50% of the families own a single vehicle, 40% of them own two, and 10% own three. No household owns more than three vehicles. The distribution is only for the illustration purpose. Users can put any specific number they have into the API tool. Besides, the number of drivers is always equal to or greater than the number of vehicle used. The function *VehicleGenerator()* is coded to deal with the generation of total number of vehicles.

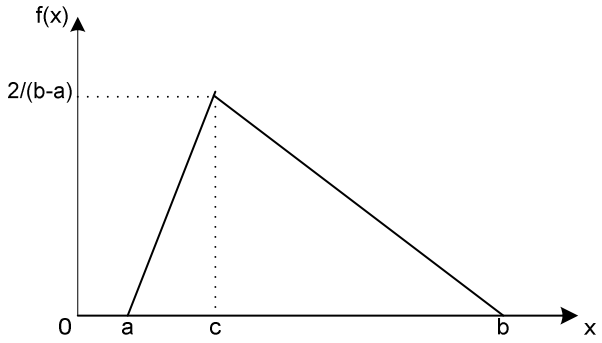
Forth, for each family's homebound attribute, in this current case that 50% families are home-bound (i.e., family-dependent), which means all vehicles in this family will be consolidated ones. The other 50% families are not family-dependent, which means all vehicles in the non-homebound family will be non-consolidated ones. The function *HomeboundGenerator()* is coded to handle the generation of family's homebound attribute. The family's homebound attribute is adjustable to demonstrate the effect of household consolidation on evacuation efficiency. For example, except for setting 50% families consolidate before evacuation, we are also interested in studying scenarios like 100% vehicles take the shortest route to safety, 100% families consolidate before evacuation, etc.

Fifth, for the distribution of departure time of the vehicles, in the context of this work, we assume it follows a triangular distribution. The triangular distribution is used since it is a

straightforward imitation of the vehicle loading process. The function *pp_start_time_assign()* is coded to set the departure time for each vehicle. A triangular random number is generated using Inverse Transform approach (Ross, 2006) as below.

Step 1. The density function is:

$$f(x)=\begin{cases}\dfrac{2(x-a)}{(b-a)(c-a)} & if\ a \le x \le c \\ \dfrac{2(b-x)}{(b-a)(b-c)} & if\ c < x \le b \\ 0 & otherwise \end{cases} \qquad (3)$$



Step 2. The distribution function is:

$$F(x)=\begin{cases}0 & if\ x < a \\ \dfrac{(x-a)^2}{(b-a)(c-a)} & if\ a \le x \le c \\ 1-\dfrac{(b-x)^2}{(b-a)(b-c)} & if\ c < x \le b \\ 1 & if\ b < x \end{cases} \qquad (4)$$

Step 3. For $X \sim Triangular(a,b,c)$, we inverted the distribution function to:

$$F^{-1}(u)=\begin{cases} a+\sqrt{u(b-a)(c-a)} & for\, u \le \dfrac{c-a}{b-a} \\ b-\sqrt{(1-u)(b-a)(b-c)} & for\, \dfrac{c-a}{b-a} > u \end{cases} \qquad (5)$$

Step 4. Generate $U \sim U(0,1)$

Step 5. If $U \le \dfrac{c-a}{b-a}$, return $X = a+\sqrt{u(b-a)(c-a)}$; Otherwise, return $X = b-\sqrt{(1-u)(b-a)(b-c)}$.

Sixth, for origin and planned destination of each vehicle, we take on the following basic distribution to generate a number of trip types. These trip types consist of different combinations of origins and destinations in the regular commute, for instance, home-to-work trip, work-to-home trip, stay-at-home trip, home-to-random (i.e. mall, post office, school, etc.) trip, work-to-random trip, and random-to-random trip. The relative percentages of each type in the basic distribution are: 50% are the vehicles who have a home-to-work trip at the beginning of the simulation, 20% have a work-to-home trip, 10% have a stay-at-home trip, 10% have a home-to-random trip, 5% have a work-to-random trip, and 5% have a random-to-random trip. This distribution is dependent on evacuation time, but for the example used, the following are assumed. The function *VehicleODGenerator()* is coded to deal with the trip generation.

At last, we put all of the generated data that contains the essential family-related and vehicle-related information into an user-defined function called *pp_init_data()*. Before

the onset of the simulation, once the network is processed, the Paramics API function *qpx_NET_postOpen()* is used to call our *pp_ini_data()* function so that our hierarchical data structure could be initialized. A brief initialization process is illustrated as figure 3-3.

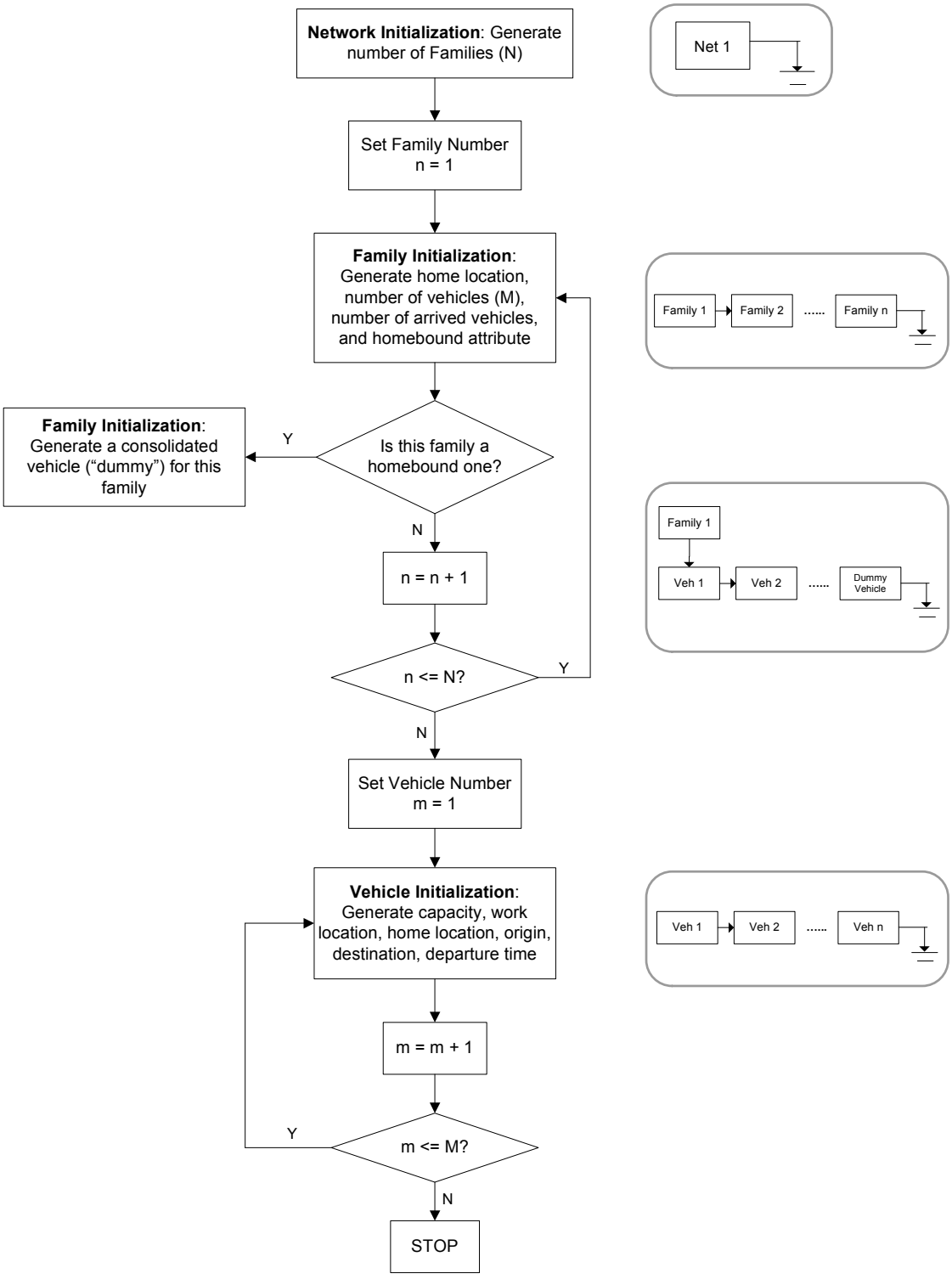


Figure 3-3. Flowchart of data initialization

3.4 Data Sorting

Once the data initialization has been completed as described in the previous section, we have the hierarchical data structure, including the family list and the vehicle list, filled with the information, like how families locate in the network, how many family members drive to work, stay at home, or drive to other locations, when each vehicle leaves for its planned destination, etc. As outlined at the beginning of this chapter, at the onset of the simulation, all vehicles are loaded into the network to simulate the regular commute traffic. That is to say, all vehicles need to be released according to their departure times.

Linked lists, as we mentioned in the section 3.2, unlike an array, only allow sequential access to elements. For example, at the beginning of the simulation, to release a vehicle #5 as show in the figure 3-4, we need to go through the whole 100 vehicles in order to decide which vehicle(s) have the departure time that equals to the current simulation time (i.e. $t=6$). Therefore, it makes sorting the vehicle list so critical before we can process the list. Since there are correlations between the family list and the vehicle list, once the vehicle list is sorted, the family list will be sorted correspondingly.

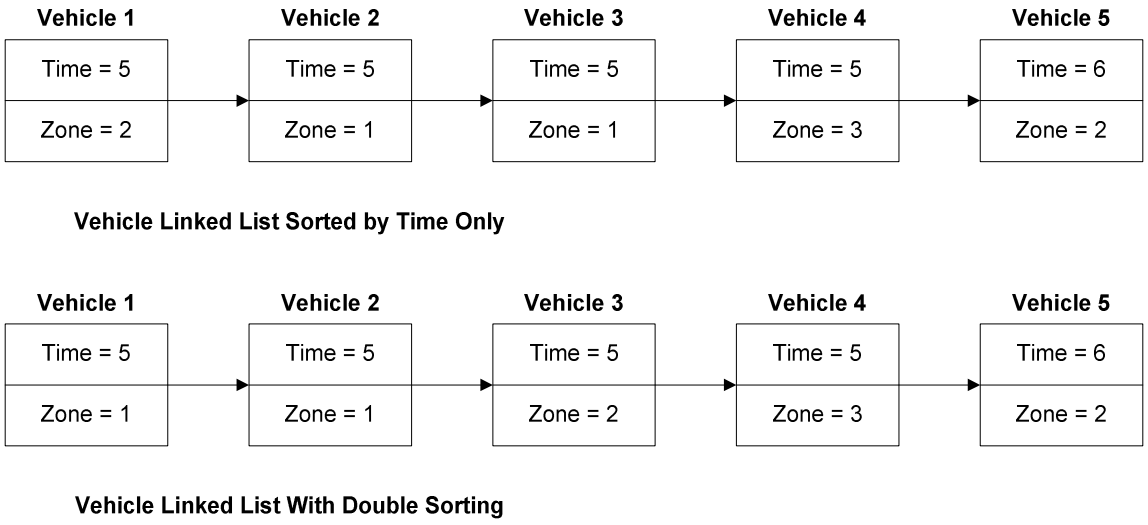


Figure 3-4. Compare of single-sorting and double-sorting for vehicle list

The purpose of sorting the vehicle structure is to release vehicles in an efficient way. To release a vehicle, in this work, the API function *qpx_ZNE_timeStep()* is used to verify if a vehicle exists whose release time equals the current simulation time in every zone. That is, in the simulation, the function *qpx_ZNE_timeStep()* is called for each zone in the network once per simulation time step. As a result, there are two concerns we need to consider in sorting the vehicle list: when to release a vehicle and where the vehicle locates. Therefore, a double-sorting process is used. We sort the vehicle list in terms of ascending vehicles' departure time first, and then by vehicles' zone index later.

The most challenging part of double-sorting a linked list is because of a linked list's sequential nature. The sorting algorithm we pick for this study is a bubble sort algorithm. It compares and swaps pair of adjacent values in an unordered list. Although it is one of the simplest sorting algorithm to understand and implement in arrays (Astrachan, 2003), it brings the complexity in sorting a linked list as we require information of previous and

next nodes of swapping nodes. A bunch of pointers are implemented in this process. Figure 3-5 shows an example about how to swap two vehicles in accordance with their departure times.

In this example, we compare vehicle #1 and #2, isolate the vehicle #2 from the list, insert it in the front of the vehicle #1, and then merge the list by linking vehicle #1 with #3. During the swap process, we have to keep track of previous and next nodes by using *"prev"* and *"tmp"* pointers. We also keep an eye on the new start of the sorted list by using *"head"* pointers. If the node being compared is not the first node, pointer *"potentialprev"* is applied. Pointer *"lst"* is used to verify the current node that will be compared with its neighbor.

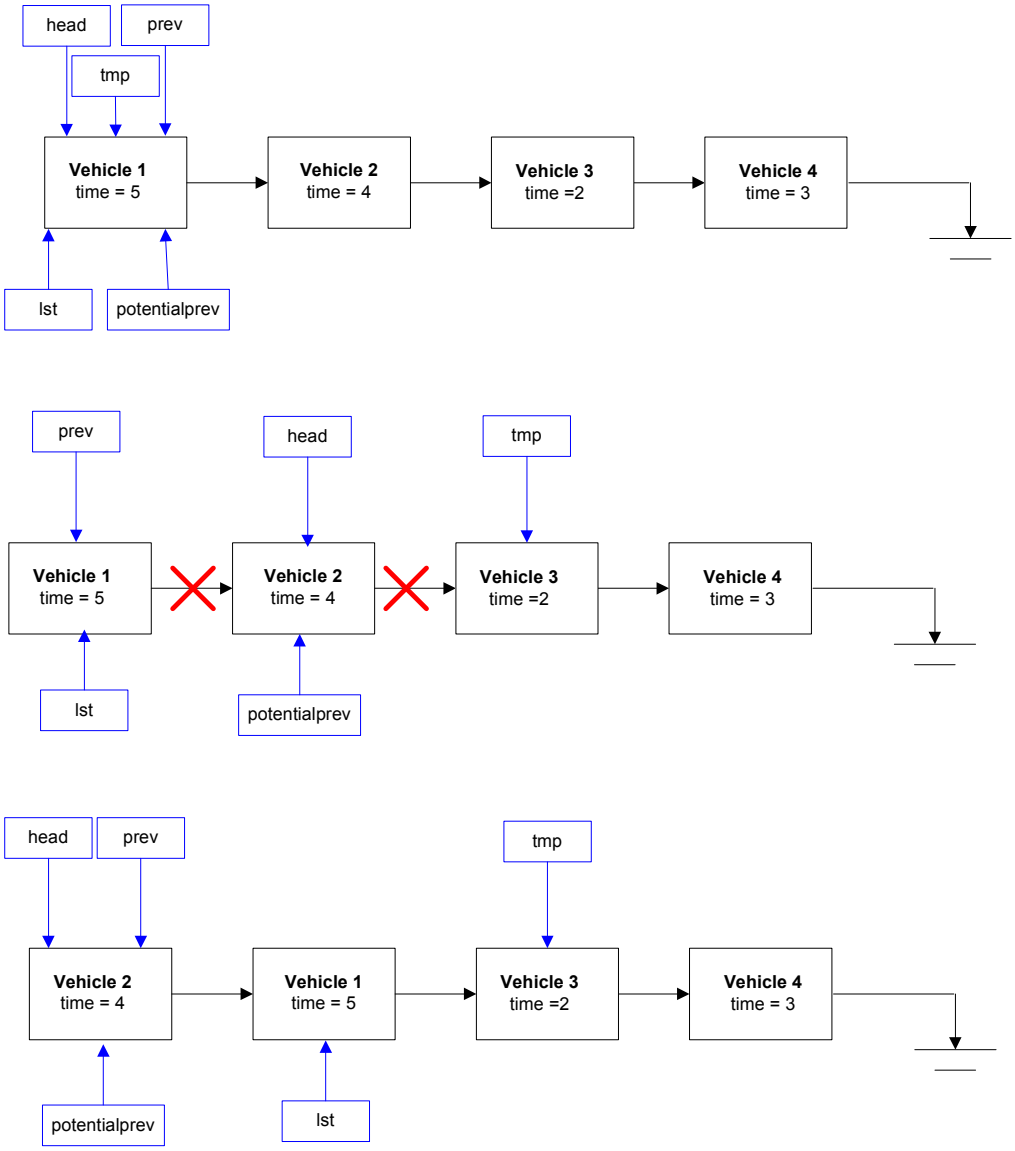


Figure 3-5. Bubble Sort a linked list

After sorting the vehicle list in terms of ascending vehicles' departure times, we repeat the same process so that the vehicles can be also ordered by increasing vehicles' zone index. After that, we code the previously described double-sorting process into an user-defined function called *pp_sort_data()*. In the same API function *qpx_NET_postOpen()* in which the *pp_init_data()* function (described in section 3.3) is

called, our function of *pp_sort_data()* is loaded. In this way, once the network is processed, after the family and vehicle data structures are initialized, the vehicles and their associated families are also be double-sorted by vehicles' departure times and zone indexes. With the initialized and sorted data structures, the regular commute traffic can be simulated as described in the following sections.

3.5 Regular Commuting Traffic Study

Once the family and vehicle data structures are built and sorted, we start the simulation with the proposed API tool by populating the network based on typical commuting traffic.

Although most of the currently available commercial traffic simulation codes are generally designed for commuter traffic simulations, a good number of them do not have the option of user-determined origins and destinations (O-Ds) for each vehicle. For example, as a widely-used microscopic traffic simulation software package, CORSIM does not have the O-D feature and can only transfer user's input turning percentages to some O-D trip possibilities. In another word, in CORSIM, vehicles move randomly in the network with respect to turning percentages that are specified by users for each intersection until they disappear at any of the possible destinations that cannot be specified by user (Zhang et al. 2003, Molina et al. 2005). Without control of independent vehicle's movement, it cannot be used in the present study.

Even for codes that do allow for individual vehicle origin-destination assignment, the criteria described in the sections 3.2 and 3.3 require that family information be available, and vehicles within a family be tracked during the simulation. Some vehicles have chained activities during the regular commute. For example, for a family member who arrived at his/her work location in the morning, his/her next stop may be the home in eight or nine hours from now. Some family members may have some random destinations, such as stopping by post office from their home to grocery stores, picking

up/dropping off their kids from schools on their way work/home, and so on. Other packages cannot do this.

One way to keep track of such information is to include the following:

(1) at the beginning of the simulation (AM/PM/...), where each family member is located (home, work, mall, …)

(2) what is the planned destination for each family member (work, home, school, …)

(3) when each family member is going to leave for his/her planned destination

(4) is there any stay-at-home family member

(5) how many family members drive to work, stay at home, or drive to other locations like school, post-office, mall, etc

(6) for those family members that finish their planned trips, does s/he have any trip chain activities

(7) what are each family member's intermediate stops if s/he has chained activities

(8) how do these activities being chained, like when s/he will leave for his/her next stop

Some of the distributions used in addressing these questions were described in the section 3.3, such as (1), (2), (3), (5), and (6), and others will be discussed later.

As presented in the section 3.2, 3.3, and 3.4, we first develop a hierarchical data structure that consists of a family list and a vehicle list, which are related to each other

(details are in the section 3.2). Next, we fill in the family and vehicle structures with information mentioned above (details are in section 3.3). Last, we double-sort the structures to make sure the vehicles are ordered in terms of their departure times and zone indexes (details are in section 3.4).

Paramics can do this study by implementing the proposed hierarchical data structure into the core simulator through the use of its API. The network-related data, such as the number of families, number of vehicles per family, etc., are added to the simulation database by using the API function *qps_NET_userdata()*. Vehicle-related data, such as vehicle capacity, origin and destination, release time, and delay time, are added by using the *qps_VHC_userdata()* function. This function is used to set the user data structure associated with the specified vehicle.

After that, with the proposed API tool, we start to simulate the typical commuting traffic. At the beginning of the simulation, all vehicles that are loaded into the network are assigned origins and destinations, like home, work, post office, mall, school, etc., as well as a departure time and the vehicle's associated home. At each simulation time step, the API function *qpx_ZNE_timestep()* is used to confirm if a vehicle exists whose departure time equals the current simulation time in every zone. If so, the vehicle is released by using our releasing function *pp_release_vehicle()*.

After all vehicles are loaded into the network, the vehicles' route choice is handled by the Paramics software. Paramics employs a route-choice heuristic based on individual

vehicle's decisions at intersections (Quadstone Limited, 2006). It considers the influence of driver's familiarity with the studied network when making route decisions. The Paramics routing procedure assures that drivers will adjust their routes dynamically based on real time traffic conditions en route. In addition, driver's aggressiveness can also be modified in Paramics. Therefore, as suggested by previous studies (Southworth 1991, Cova and Johnson 2002, Chena and Zhen 2008), Paramics is generally considered as an appropriate microscopic simulation model for evacuation analysis. Figure 3-5-1 demonstrates the animation of the regular traffic in a simulated environment. The yellow dots represent the vehicles with the user-defined data structure. The green line represents the arterial streets while red lines for the freeways.

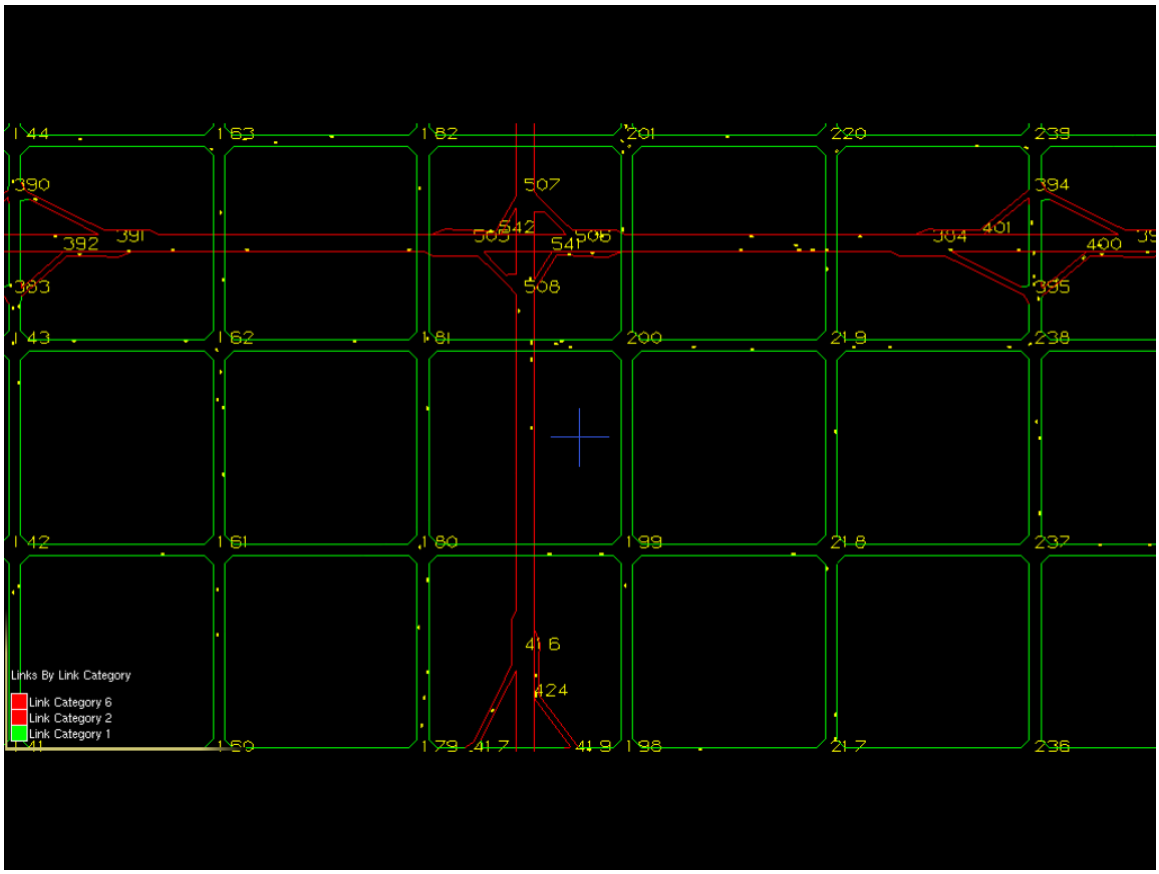


Figure 3-6-1. Demonstration of the regular simulation

Last but not least, an emergency event is initiated at a predetermined time, which interrupts the regular commuting traffic. A few modules are developed and loaded to study different kinds of driver behaviors under this situation. They include vehicles who have not yet been released into the network, vehicles that may realize there is an evacuation while they are sill en route, vehicles who arrive at their destinations affected/unaffected by the evacuation, and vehicles who have consolidated household processes. A detailed discussion of these modules is presented in the following section 3.6, section 3.7, section 3.8 and section 3.9.

.

3.6 Sub-Model of Yet To Be Released Vehicles

As described previously, in the context of this research, the simulation starts during the regular commuting traffic. Then an evacuation event is initiated after the network is reasonably populated. At this moment, some vehicles who have already been assigned the origins and destinations in the initialization process are waiting for their turns to be released into the network. However, when the order to evacuation is given, these vehicles' departure times have not been reached yet. Therefore, we name this kind of vehicles as yet to be released (YTBR) vehicles. Besides, thanks to the evacuation, the behaviors of such vehicles are changed. One of the examples is, for a vehicle that is still at its home and plans to go to work half an hour later, once it is aware of the evacuation, this vehicle, say a non-consolidated one, will go to the closest shelter as soon as possible. Therefore, we develop this sub-model to study and model the behaviors that how those YTBR vehicles react to the evacuation.

A great deal of the previous work on emergency evacuation study evacuees' reaction by assuming that all those evacuees depart immediately once the evacuation starts. However, these studies ignore the delay time between the evacuation starts and each of individuals finds about the evacuation. In this study, we propose a more realistic assumption that an awareness time for each evacuee may differ. This realization may depend on the efficiency of evacuation information dissemination. The awareness time may vary by each person's location, his/her planned destination, his/her current activity, and other issues such as whether or not the person has to consolidate with other family members.

For each YTBR vehicle, its current location and planned destination composes different types of trips. In this study, we explore six typical trip patterns that have observed in a regular traffic condition. These trip types include: home-to-work, work-to-home, home-to-random (i.e. school, post office, mall, etc.), work-to-random, stay-at-home, and random-to-random. These trip types have a significant effect on the evacuees' behaviors in the network, which contributes a lot to the traffic congestions during the evacuation. For instance, if a YTBR vehicle has a work-to-home trip, which means this vehicle is currently at work and is planning to go home a few hours later, once it finds out there is an evacuation, this vehicle may leave for the closest shelter as soon as possible, referring to Table 3-2. Figure 3-5-2 is a summarized version of the table 3-2.

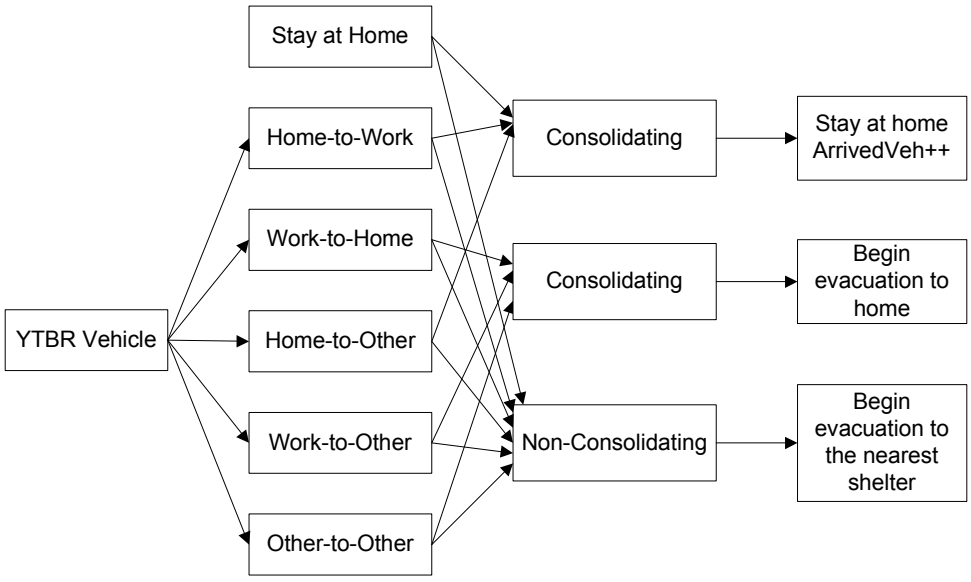


Figure 3-5-2. Summary of the scenarios for the YTBR vehicles

Another significant feature of the developed API tool is to model drivers' consolidated household evacuation behavior. In the example above, if this work-to-home YTBR vehicle is a consolidated household vehicle, once it is aware that the evacuation starts, instead of going to the shelter immediately, it will return to its home first. Then after all vehicles associate with its family arrive, they then evacuate together in a single vehicle, as shown in Table 3-2 and the figure 3-5-2.

Once we have the YTBR vehicle's trip type and consolidated attribute identified, in the proposed approach, we check whether or not this vehicle's planned departure time earlier than the evacuation start time plus its awareness time. Different treatments are made according to the vehicle's current location and expected destination. The flowchart depicting the logic implemented for this sub-model is shown in Figure 3-6. In line with this flowchart, a list of combinations of each YTBR vehicle's current location, planned destination, consolidation attribute, awareness of the evacuation, current activity, scheduled changes, and special tips, are summarized in the table 3-2, which leads to a set of 36 possible scenarios that can be simulated by this sub-model.

Among those scenarios, there are some special cases we need to pay attention to. One particular case is when the departure time of an YTBR vehicle is just minutes from the onset of the evacuation. This trip's start time should not be inferred by the evacuation event, but the vehicle will be assigned an awareness time, which will be used in the sub-model of the en route vehicles. Further details are explained in the section 3.7.

Another example of the 36 possible scenarios is for a stay-at-home vehicle, a special vehicle type (i.e. vehType = 1, as shown in Table 3-2) and a large enough departure time (i.e. StartTime=999,999 seconds) need to be assigned to prevent this kind of vehicle being released from the queue of the sorted vehicles at the beginning of the simulation.

We implement this sub-model by using a Paramics API function qpx_ZNE_timeStep(). This function is called for each zone in the network at each simulation time step. There is another API function qpx_NET_timeStep(), which is called for the whole network once per simulation step. Both of these functions are suitable for implementing our sub-model of YTBR vehicles. The reason we pick the former is that, per our test, the function qpx_ZNE_timeStep() is called earlier than the function qpx_NET_timeStep() during a simulation, which leads to an increase of the computation efficiency. After the implementation, during the simulation process, at the moment the evacuation starts, each yet to be released vehicle will be scanned, and then the corresponding change will be applied to the vehicle following the modeling flowchart (i.e. figure 3-6).

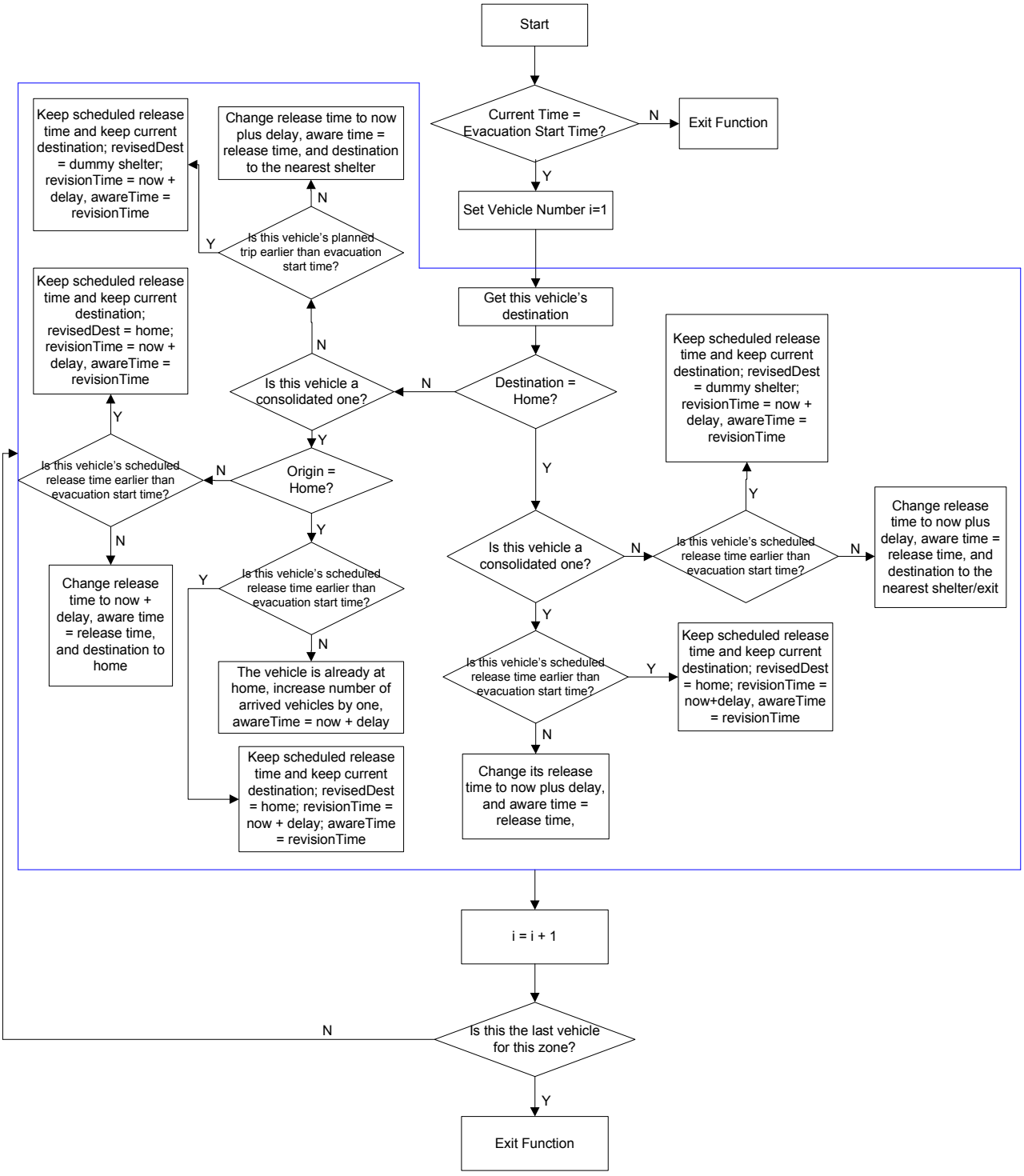


Figure 3-7. Simulation of Yet To Be Released Vehicles (API Function: qpx_ZNE_timeStep())

Table 3-2. Summary of Yet to be Released Vehicles' Behaviors

Scenarios	Current location	Planned destination	Consolidate(?)	Planned trip start time	Action	Release from current location	New destination	Awareness time	Notes
1	Home	Home	Yes	>evac time	arrived veh++	No	--	incidence + delay	veh type 1 can be used
2	Home	Home	No	>evac time	begin evac	Yes	virtual shelter	incidence + delay	veh type 1 can be used
3	Home	Home	Yes	<evac time					treat as 1
4	Home	Home	No	<evac time					treat as 2
5	Home	Work	Yes	>evac time	arrived veh++	No	--	incidence + delay	veh type 1 can be used
6	Home	Work	No	>evac time	begin evac	Yes	virtual shelter	incidence + delay	
7	Home	Work	Yes	<evac time	change dest en route	as planned	home	incidence + delay	
8	Home	Work	No	<evac time	change dest en route	as planned	virtual shelter	incidence + delay	
9	Home	Other loc	Yes	>evac time	arrived veh++	No	--	incidence + delay	veh type 1 can be used
10	Home	Other loc	No	>evac time	begin evac	Yes	virtual shelter	incidence + delay	
11	Home	Other loc	Yes	<evac time	change dest en route	as planned	--	incidence + delay	
12	Home	Other loc	No	<evac time	change dest en route	as planned	virtual shelter	incidence + delay	
13	Work	Home	Yes	>evac time	begin evac	Yes	--	incidence + delay	
14	Work	Home	No	>evac time	begin evac	Yes	virtual	incidence +	

							shelter	delay	
15	Work	Home	Yes	<evac time		as planned	--	incidence + delay	
16	Work	Home	No	<evac time	change dest en route	as planned	virtual shelter	incidence + delay	
17	Work	Work	Yes	>evac time					Assume no worker will stay at work
18	Work	Work	No	>evac time					Assume no worker will stay at work
19	Work	Work	Yes	<evac time					Assume no worker will stay at work
20	Work	Work	No	<evac time					Assume no worker will stay at work
21	Work	Other loc	Yes	>evac time	begin evac	Yes	home	incidence + delay	
22	Work	Other loc	No	>evac time	begin evac	Yes	virtual shelter	incidence + delay	
23	Work	Other loc	Yes	<evac time	change dest en route	as planned	home	incidence + delay	
24	Work	Other loc	No	<evac time	change dest en route	as planned	virtual shelter	incidence + delay	
25	Other loc	Home	Yes	>evac time	begin evac	Yes	--	incidence + delay	
26	Other	Home	No	>evac time	begin evac	Yes	virtual	incidence +	

	loc						shelter	delay	
27	Other loc	Home	Yes	<evac time		as planned	--	incidence + delay	
28	Other loc	Home	No	<evac time	change dest en route	as planned	virtual shelter	incidence + delay	
29	Other loc	Work	Yes	>evac time	begin evac	Yes	home	incidence + delay	
30	Other loc	Work	No	>evac time	begin evac	Yes	virtual shelter	incidence + delay	
31	Other loc	Work	Yes	<evac time	change dest en route	as planned	home	incidence + delay	
32	Other loc	Work	No	<evac time	change dest en route	as planned	virtual shelter	incidence + delay	
33	Other loc	Other loc	Yes	>evac time	begin evac	Yes	home	incidence + delay	
34	Other loc	Other loc	No	>evac time	begin evac	Yes	virtual shelter	incidence + delay	
35	Other loc	Other loc	Yes	<evac time	change dest en route	as planned	home	incidence + delay	
36	Other loc	Other loc	No	<evac time	change dest en route	as planned	virtual shelter	incidence + delay	

3.7 Sub-Model of En Route Vehicles

In the aforementioned sub-model of yet to be released vehicles, we propose an API tool to simulate the behaviors of those vehicles that have not yet been released into the network at the beginning of the evacuation. At the commencement of an emergency evacuation, the network should be loaded with commute traffic, i.e. each one is in a trip from its origin to its destination. We call these vehicles en route (ENR) vehicles. Behaviors of these vehicles are also changed due to the evacuation. One of the examples of such changed behaviors is that, for a vehicle in the middle of a trip from home to work, once the driver is aware of the evacuation, this vehicle that we assumed is a consolidating one will change its destination dynamically en route, namely, this vehicle will go back home immediately instead of going to work as scheduled. The purpose of this sub-model of en route vehicles is to extend the proposed API tool to model how each of ENR vehicles react to the evacuation.

One of the most significant features of this study is to add the capability by using an API to make Paramics be capable of reacting to dynamic changes in an en route vehicle's destination via the proposed API tool. As mentioned in section 3.5, a great deal of existing traffic simulation tools does not have the capability to track individual vehicle's destination. For those that have the options of user-defined origins and destinations, based on our knowledge, none can model the dynamic change of an en route vehicle's destination during a simulation. In the proposed tool, we model this kind of dynamic change of destinations by using revision time and revised destination. These definitions are explained in the following example. For a consolidated vehicle who is in a trip from

its home to work when the evacuation starts, the vehicle will continue on its original route to work. A new destination home for consolidating vehicle and shelter for non-consolidating vehicle is assigned and a start time – current time a random time is given. We call this new destination as the revised destination and the random delay time as the revision time. The revision time and revised destination may vary by each ENR vehicle's location, planned destination, and current activity. When the revision time is reached, this vehicle will travel towards its new destination. A figure that summarizes this and other cases of ENR vehicles is shown below.

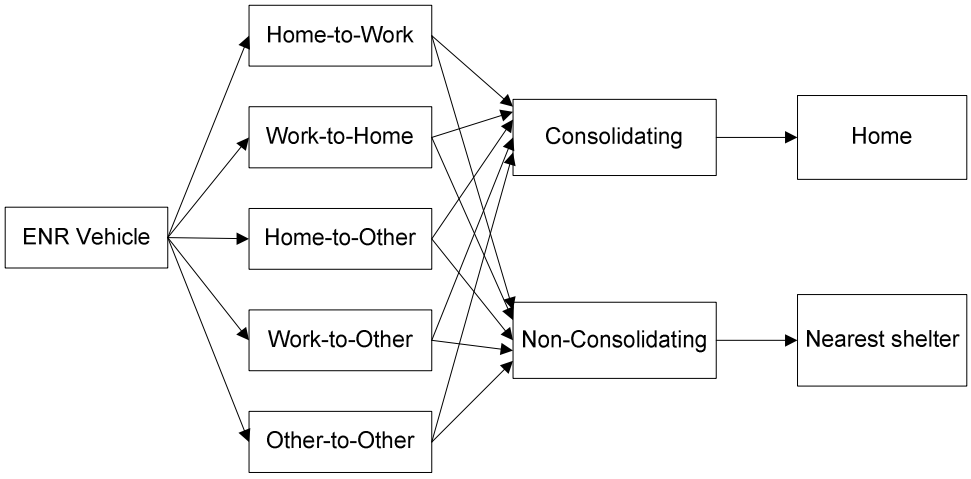


Figure 3-5-3. Summary of the scenarios for the ENR vehicles

Another feature of this sub-model is, like in the sub-model of yet to be released vehicles, for each ENR vehicle, it is also associated with a specific trip type. There are five typical trip types we studied in this sub-model: home-to-work, work-to-home, home-to-other location (i.e. school, post office, mall, etc), work-to-other location, and other location-to-other location. Unlike the YTBR vehicles we studied in the previous section,

we do not need to study those vehicles that have the stay at home/work trips as these kinds of trips are not applicable to en route vehicles.

In the concept of household consolidation study, this extended API tool can also model each ENR driver's consolidated household evacuation behavior. In the example above, if the en route home-to-work vehicle is a non-consolidated one, which means it will go to the closest shelter as soon as possible once it is aware of the evacuation, we assign its revised destination as a virtual shelter rather than its home location. Its revision time will still be the same. The reason why we use a "virtual shelter" is because we assume that when evacuation starts, each non-consolidated vehicle will go to the closest shelter based on its current location. However, at the moment we assign each ENR vehicle a revised destination, we do not know which shelter is the closest one. In another word, since this vehicle is still traveling to its originally planned destination, the closest shelter we pick at that time may be different from the closest one that responds to the vehicle's location at the revision time. For that reason, we use a virtual shelter as a temporary revised destination, and afterward when the revision time is reached, the virtual shelter will be replaced by the closest shelter that the vehicle will move towards at that moment.

As mentioned in the previous section, the proposed API tool presents a more accurate approach by taking each individual vehicle's reaction time to the evacuation into consideration. In this sub-model, for each ENR vehicle, in most cases, its revision time and awareness time, which is the duration between the onset of the evacuation and the

time the driver comes aware of the evacuation, are identical. In a few cases, the revision time and the awareness time of one vehicle may not be the same. For instance, for a consolidating, work-to-home ENR vehicle, as its current route is identical to its planned evacuation route, there is no need for this vehicle to be assigned a revised destination. Therefore, this vehicle's revision time is void while its awareness time is equal to the evacuation time plus some delay that models how the evacuation information is disseminated.

Another special case we need to consider is that an ENR vehicle is seconds from its current destination, but its awareness/revision time is much greater. In this case, this vehicle will finish its current trip as well as being assigned a revised destination and an awareness time. Afterward, once the vehicle arrives at its destination, it will be handled in the sub-model of arrived vehicles by generating a new trip in which the origin will be its current destination, the destination will be its revised destination, and the release time will be its awareness time. A detailed process will be explained in section 3.8.

Overall, in this sub-model of ENR vehicles, we model the dynamic changes of an en route vehicle's destination during a simulation, the traffic maneuvers of different trip types of ENR vehicles, the ENR vehicle's consolidated household behaviors, and the ENR vehicle's revision times, revised destinations and awareness times. The flowchart depicting the logic implemented for this sub-model of en route vehicles is shown in Figure 3-7. Furthermore, according to the previous discussions, in general, there are a set of 28 possible scenarios that combine each ENR vehicle's trip type, its consolidation

attribute, its revised destinations, its awareness times, and the relevant action. These scenarios are summarized into table 3-2.

We implement this sub-model by using an API function qpx_LNK_vehicleTimeStep(). This function is called to scan each vehicle in each link of the network once per simulation step. A similar API function is qpx_LNK_timeStep(), which is called to examine each link of the network once per simulation step. The reason we do not pick qpx_LNK_timeStep() is that, in this function, all vehicles, no matter in which state of yet-to-be-released, en-route or arrived-at-its-destination, are all involved. In contrast, the function qpx_LNK_vehicleTimeStep() only deals with those vehicles that are currently on a link, namely, those vehicles that are currently in the middle of a trip. Therefore, the use of function qpx_LNK_vehicleTimeStep() leads to high computing efficiency and a more concise and accurate modeling process compared to other API functions.

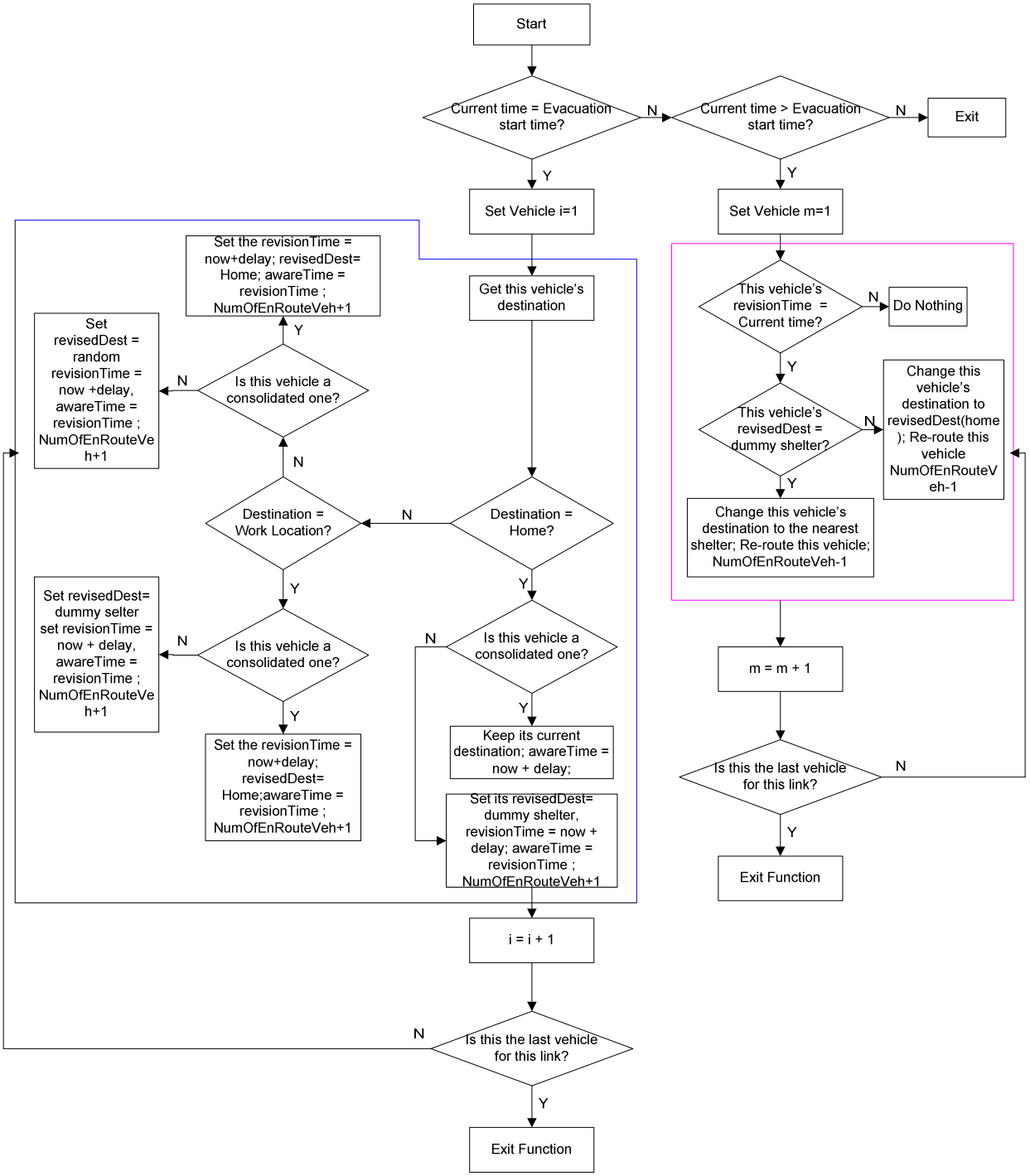


Figure 3-8. Simulation of En Route Vehicles (API Function: qpx_LNK_vehicleTimeStep())

Table 3-3. Summary of En Route Vehicles' Behaviors

Scenarios	Origin	Destination	Consolidate (?)	Trip start time	Action	Revised destination	Revision time	Notes
1	Home	Work	Yes	>evac time	re-route this veh	home	as planned	Number of en route vehicle -1
2	Home	Work	No	>evac time	re-route this veh	nearest shelter	as planned	Number of en route vehicle -1
3	Home	Work	Yes	"=evac time"	assign revised time and destination	home	now + delay	Number of en route vehicle +1
4	Home	Work	No	"=evac time"	assign revised time and destination	virtual shelter	now + delay	Number of en route vehicle +1
5	Home	Other loc	Yes	>evac time	re-route this veh	home	as planned	Number of en route vehicle -1
6	Home	Other loc	No	>evac time	re-route this veh	nearest shelter	as planned	Number of en route vehicle -1
7	Home	Other loc	Yes	"=evac time"	assign revised time and destination	home	now + delay	Number of en route vehicle +1
8	Home	Other loc	No	"=evac time"	assign revised time and destination	virtual shelter	now + delay	Number of en route vehicle +1
9	Work	Home	Yes	>evac time	re-route this veh	as planned	as planned	Number of en route vehicle -1
10	Work	Home	No	>evac time	re-route this veh	nearest shelter	as planned	Number of en route vehicle -1
11	Work	Home	Yes	"=evac time"	as planned	--	--	--
12	Work	Home	No	"=evac time"	assign revised time and destination	virtual shelter	now + delay	Number of en route vehicle +1
13	Work	Other loc	Yes	>evac time	re-route this veh	home	as planned	Number of en route vehicle -1

14	Work	Other loc	No	>evac time	re-route this veh	nearest shelter	as planned	Number of en route vehicle -1
15	Work	Other loc	Yes	"=evac time"	assign revised time and destination	home	now + delay	Number of en route vehicle +1
16	Work	Other loc	No	"=evac time"	assign revised time and destination	virtual shelter	now + delay	Number of en route vehicle +1
17	Other loc	Home	Yes	>evac time	re-route this veh	as planned	as planned	Number of en route vehicle -1
18	Other loc	Home	No	>evac time	re-route this veh	nearest shelter	as planned	Number of en route vehicle -1
19	Other loc	Home	Yes	"=evac time"	as planned	--	--	--
20	Other loc	Home	No	"=evac time"	assign revised time and destination	virtual shelter	now + delay	Number of en route vehicle +1
21	Other loc	Work	Yes	>evac time	re-route this veh	home	as planned	Number of en route vehicle -1
22	Other loc	Work	No	>evac time	re-route this veh	nearest shelter	as planned	Number of en route vehicle -1
23	Other loc	Work	Yes	"=evac time"	assign revised time and destination	home	now + delay	Number of en route vehicle +1
24	Other loc	Work	No	"=evac time"	assign revised time and destination	virtual shelter	now + delay	Number of en route vehicle +1
25	Other loc	Other loc	Yes	>evac time	re-route this veh	home	as planned	Number of en route vehicle -1
26	Other loc	Other loc	No	>evac time	re-route this veh	nearest shelter	as planned	Number of en route vehicle -1
27	Other loc	Other loc	Yes	"=evac time"	assign revised time and destination	home	now + delay	Number of en route vehicle +1
28	Other loc	Other loc	No	"=evac time"	assign revised time and destination	virtual shelter	now + delay	Number of en route vehicle +1

As shown in the figure 3-7, for each vehicle in each link of the network once per simulation step, the function *qpx_LNK_vehicleTimeStep()* checks if the vehicle is en route once the evacuation starts. If so, based on this vehicle's current location, planned destination, household attribute, revised destination, and revision time, the relevant changes will be implemented. Another API function *qps_VHC_destination()* is also used to set the new destination zone for those vehicles that have to change their destinations if necessary while they are still en route to their originally scheduled destination.

Next a group of figures demonstrates the simulation of the ENR vehicles after the implementation.

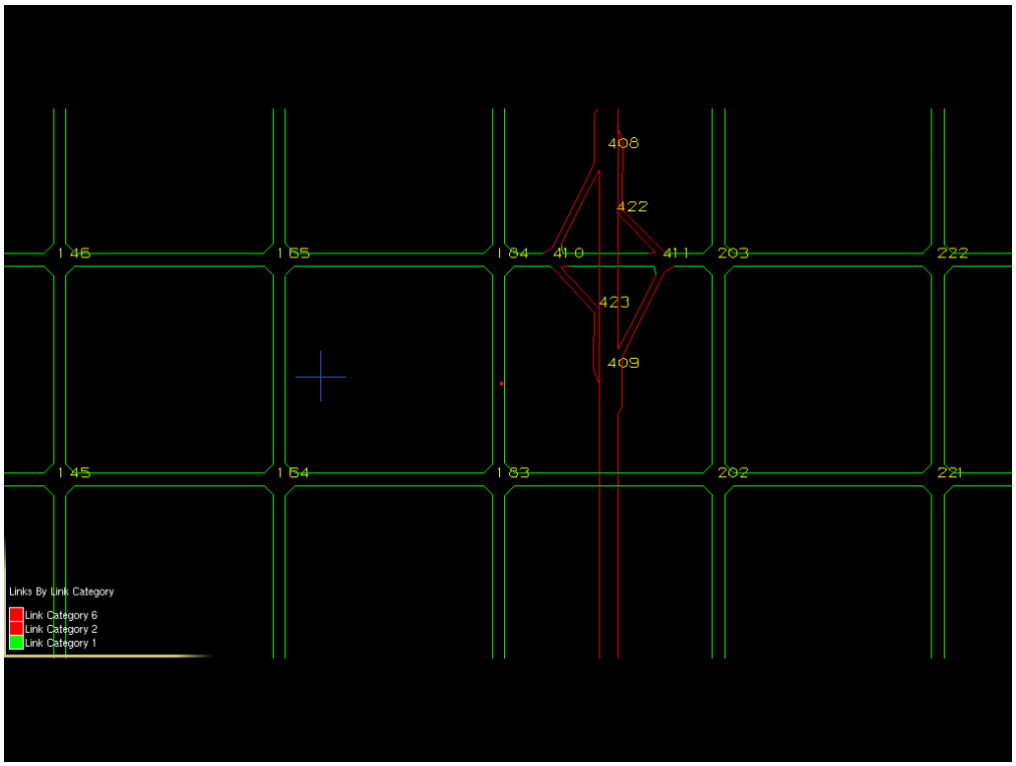


Figure 3-9-4(a). Demonstration of the en route vehicle simulation

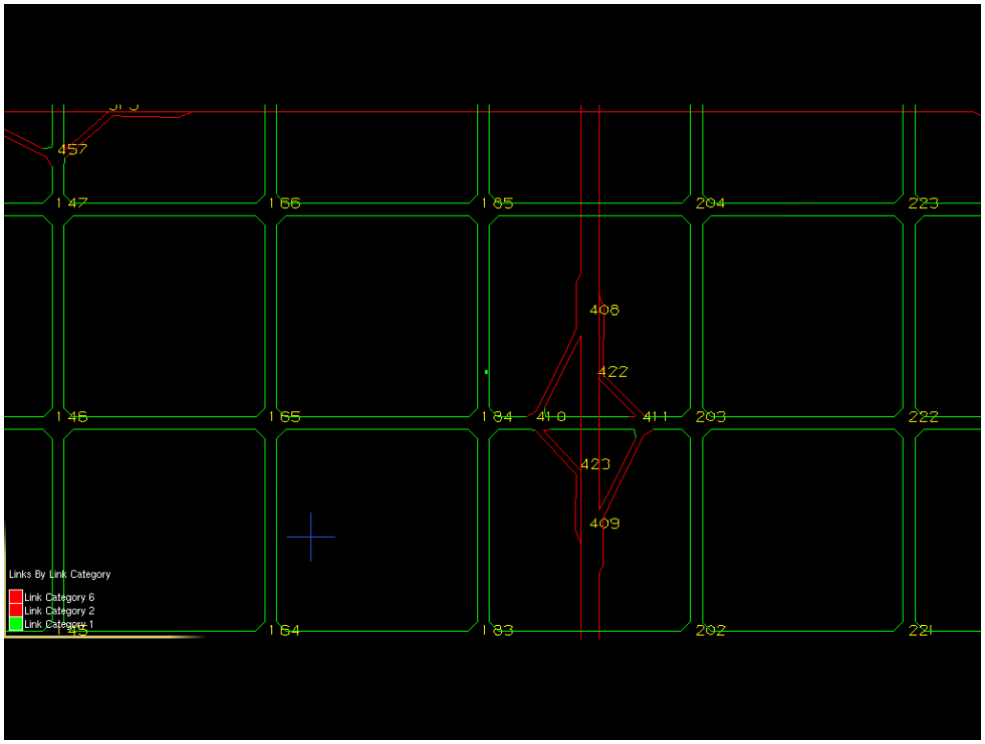


Figure 3-10-4(b). Demonstration of the en route vehicle simulation

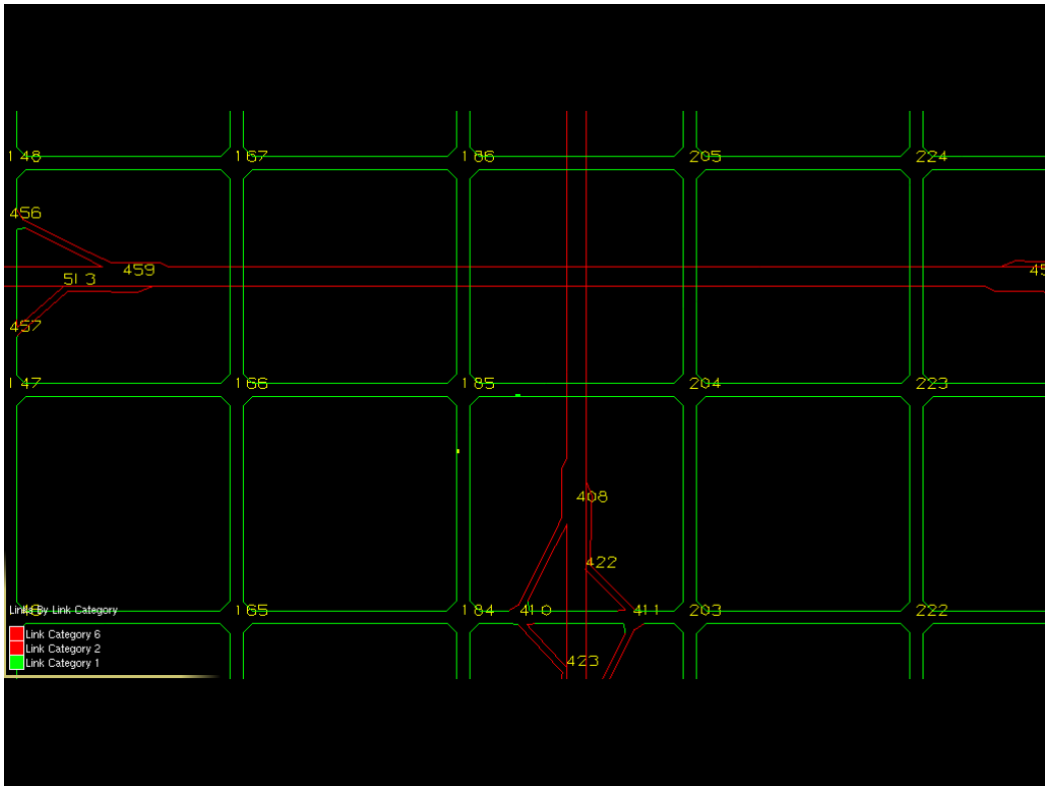


Figure 3-11-4(c). Demonstration of the en route vehicle simulation

3.8 Sub-Model of Arrived Vehicles (Vehicle Consolidated Process Included)

In the previously sections, we proposes an API tool to model the regular commute traffic, to simulate the behaviors of those vehicles that have not yet been released into the network due to the evacuation, and to model how a vehicle changes its destination dynamically while it is still en route when the evacuation starts. Finally, for all these vehicles that are loaded into the network during the simulation, some of them may arrive at their destinations before the start of the evacuation. Others may arrive at their destinations after the evacuation. We named all these vehicles arrived (ARV) vehicles. Some of these arrived vehicles even have chained activities. For instance, for a vehicle that has just arrived at its work location in the morning, its next stop may be its home location in eight or nine hours. Thus, in this section, we propose a sub-model of arrived vehicles, which models the behaviors, especially, the sequence of behaviors of each arrived vehicle in the simulation process.

For each arrived vehicle, the destination it reaches may be its final destination or just be one of its intermediate stops. Most of the existing simulation software packages cannot model vehicles' chained behaviors, such as TSIS/CORSIM (ITT Industries, 2000). Even for a limited number of traffic simulation tools, like VISSIM, DYNASMART, which are able to simulate vehicles' trip chain activities, the way they handle such behaviors has some limitations. The significant difference is that these tools can only do the vehicle assignment at the beginning of the simulation, while the proposed API tool can do the vehicle assignment in the middle of the simulation.

For example, in VISSIM, to model a trip-chain vehicle's movement, user has to fulfill the chained trip information into a specific file. The information includes the vehicle number, vehicle type, origin zone number, departure time, intermediate destination zone number(s), and stay time. DYNASMART has the similar requirement but with separated input files, such as zone data file, O-D demand matrix file, generation link file, destination node file, vehicle loading file, and path file. We use an example to illustrate how these tools work. For instance, there is a vehicle with a scheduled home-work-home trip. These tools can simulate such kind of movement by pre-assigning the vehicle's intermediate destination, i.e. its work location. However, if there is an emergency evacuation, this vehicle may not travel to its work destination as scheduled, but to the nearest shelter as soon as it finds out the threat (i.e. it is a non-consolidated vehicle). In this case, based on our knowledge, none of the existing simulation tools is capable to re-assign the new destination to the vehicle in the middle of a simulation. However, as one of the most significant features of this study, our proposed API tool can teach Paramics to be able to do vehicles' intermediate destination re-assignment during a simulation.

To model such kinds of scheduled and unscheduled trip-chaining behaviors, in this tool, a set of dummy vehicle is used. This type of dummy vehicles is generated during the data initiation process with a very large departure time (i.e. $t = 2{,}000{,}000$ seconds) which prevents this type of vehicles from being sorted and released at the beginning of the simulation. Therefore, the dummy vehicles are reserved to create new trips during the chained events. During the simulation, once a vehicle arrives at its destination, in Paramics, this vehicle disappears at its destination. However, with our proposed API

tool, at this moment, the dummy vehicle is used to take over the "disappeared" one. Information carried by the "disappeared" vehicle, like its home location, its homebound attribute, its current location, etc, is passed on to this "dummy" vehicle. After that, this "dummy" vehicle is being assigned a new trip's information that comes from the potential chain movements of the "disappeared" one. In this way, a new trip is created in the middle of the simulation.

As shown in the example above, if the arrived, home-to-work vehicle has another work-to-home trip eight hours from now, then for this "dummy" vehicle that will replace the arrived vehicle, it has the arrived vehicle's work location as its origin, the arrived vehicle's home location as its destination, and eight hours from now as its departure time. The flowchart depicting the logic implemented for this sub-model of arrived vehicles is shown in figure 3-8. After that, thanks to the flexibility of the hierarchical data structure that we described in the section 3.2, we can insert this "dummy" vehicle that holds the information for the new trip into the proper location of the sorted vehicles. This kind of insertion process will be explained in more details in section 3.9.

Furthermore, for those arrived vehicles that are affected by the evacuation, there is a special case we have to consider. In this case, the arrived vehicle is not aware of the evacuation that happened only minutes before the vehicle reached its destination. As described in the previous section 3.7, this vehicle has already been assigned a revised destination and a revision time when it was en route. However, in this case, the revision time is not reached when the vehicle arrives at its destination. Therefore, for such type of

arrived vehicles, when we use the “dummy” vehicle to replace the arrived one, we assign the arrived vehicle’s revised destination as the “dummy” vehicle’s destination, and the arrived vehicle’s revision time as its departure time. Both the revised destination and revision time are identified in the sub-model of en route vehicles (i.e. section 3.7).

Some readers may raise the question that if we could treat this “dummy” vehicle as a yet to be released (YTBR) vehicle, since it is a vehicle that has not been released to the network yet. The answer is no. For the type of YTBR vehicles, as discussed in section 3.6, at the onset of the evacuation, each YTBR vehicle will be assigned a revised destination, an awareness time and a new departure time according to each vehicle’s attributes. After that, the whole vehicle list will be re-sorted in terms of each vehicle’s departure time and departure zone. However, for the type of arrived vehicles, once a vehicle arrived at its destination, this vehicle disappears from the network, and a new “dummy” vehicle will substitute it for continuing its subsequent trip(s). A new trip, including a new destination, an awareness time and a new departure time, is generated for this “dummy” vehicle, and then this “dummy” vehicle will be inserted into the current sorted vehicle list (i.e. this kind of insertion process is explained in section 3.9). Comparing with the YTBR vehicles, this procedure avoids double-sorting the vehicle list each time when an arrived vehicle is replaced by the dummy vehicle. In this way, the total time and efforts spending on the computation are saved and the overall efficiency is improved.

Like in other sub-models, each arrived vehicle is also associated with a specific trip type. The types of trips we study in this model include: home-to-work, work-to-home, home-to-other location (i.e. school, post office, mall, etc.), work-to-other location, other location-to-home, other location-to-work, other location-to-other location.

In the sub-model of arrived vehicles, we also model the arrived vehicles' evacuation dynamics by checking whether it is a consolidating vehicle. If the arrived vehicle is a non-consolidating vehicle, it will go to the closest shelter once it is aware of the evacuation. Otherwise, if the arrived vehicle is a consolidated one, it will go to its consolidation point (i.e. its home) to meet with other family members. If the destination that the consolidated vehicle just arrived at is exactly its consolidation point, this vehicle will be involved in a household consolidation process, which is explained in more details in the following section 3.9.

Overall, in this sub-model of arrived vehicles, for each vehicle that has arrived at its destination, it may have scheduled succeeding destination(s) in a regular commute situation, or it may have unscheduled trip-chain movement(s) in an emergency evacuation condition. In this sub-model, we model both kinds of trip-chain movements during a simulation by using our proposed API tool. We also simulate the different trip types of the arrived vehicles, and the household behaviors of the arrived vehicles. Generally, there are a set of 16 possible of scenarios that combine each arrived vehicle's characteristics, like its current location, planned destination, revision time, revised

destination, awareness of the evacuation, household attribute, which is summarized into the table 3-4.

This sub-model is implemented by using a Paramics API function *qpx_VHC_arrive()*, which is called once a vehicle arrives at its destination zone. As shown in the figure 3-8, each arrived vehicle is scanned and then being modified correspondingly, like being replaced by a “dummy” vehicle for generating a subsequent trip.

We also have a group of figures demonstrate the simulation of the consolidation approach after the implementation, as shown in the figure 3-8 (a) – (d).

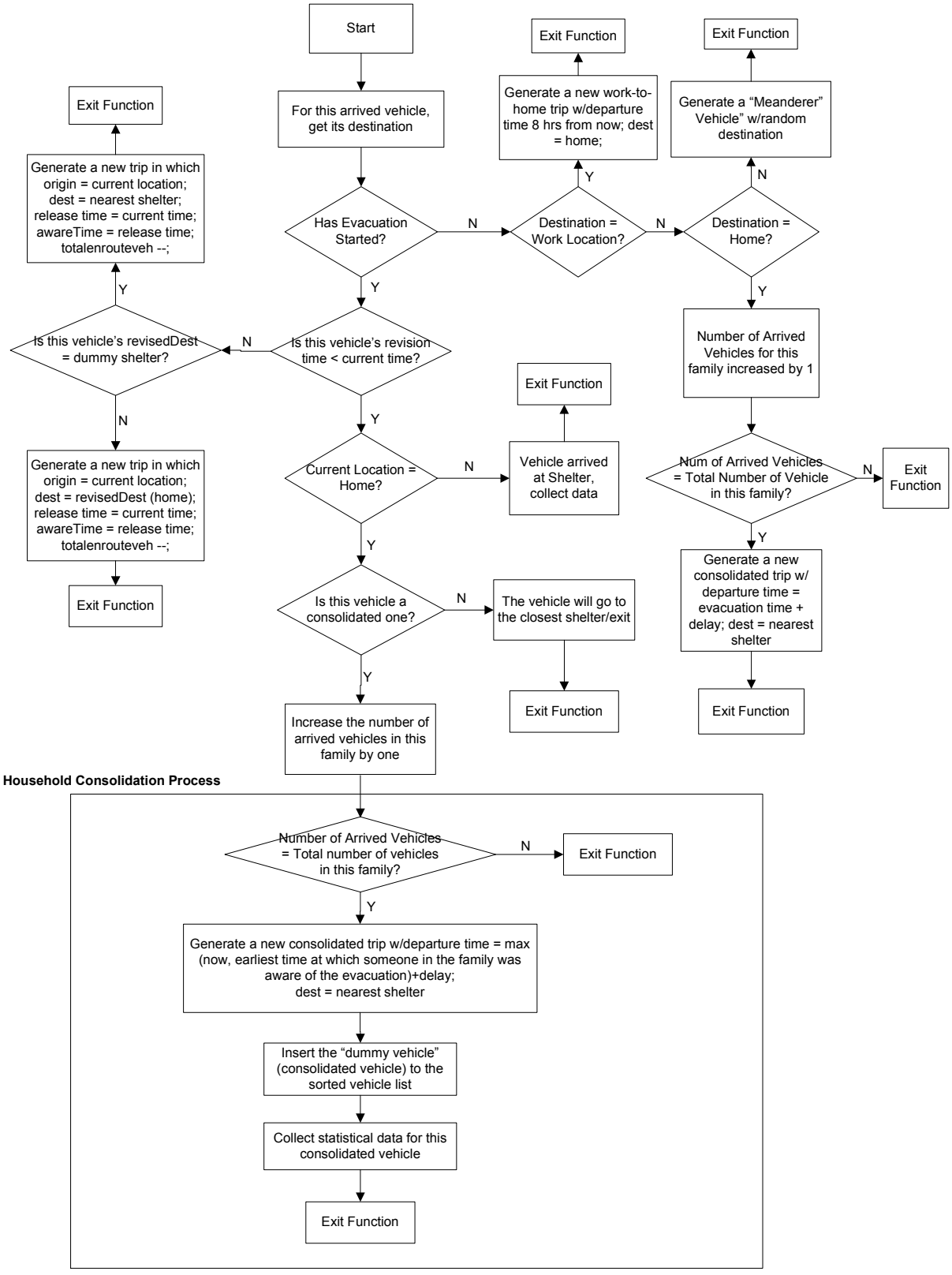


Figure 3- 12. Simulation of Arrived Vehicles (API Function: qpx_VHC_arrive())

Table 3-4. Summary of Arrived Vehicles' Behaviors

Scenarios	Previous location	Arrived destination	Consolidate(?)	Arrival time	Action	New destination	Departure time	Notes
1	Home/Other loc	Work	Yes	>evac time	generate a new trip	home	current time	Number of en route vehicle -1
2	Home/Other loc	Work	No	>evac time	generate a new trip	nearest shelter	current time	Number of en route vehicle -1
3	Home/Other loc	Work	Yes	<evac time	generate a new trip	home	8 hrs from now	--
4	Home/Other loc	Work	No	<evac time	generate a new trip	other loc	8 hrs from now	--
5	Home/Work/ Other loc	Other loc	Yes	>evac time	generate a new trip	home	current time	Number of en route vehicle -1
6	Home/Work/ Other loc	Other loc	No	>evac time	generate a new trip	nearest shelter	current time	Number of en route vehicle -1
7	Home/Work/ Other loc	Other loc	Yes	<evac time	generate a new trip	home	now + delay	--
8	Home/Work/ Other loc	Other loc	No	<evac time	generate a new trip	other loc	now + delay	--
9	Work/Other loc	Home	Yes	>evac time	start consolidation process	nearest shelter	consolidation trip depature time	arrived veh +1
10	Work/Other loc	Home	No	>evac time	begin evac	nearest shelter	now + delay	--
11	Work/Other loc	Home	Yes	<evac time	--	--	--	arrived veh +1
12	Work/Other loc	Home	No	<evac time	generate a new trip	other loc	now + delay	--
13	Work/Home/ Other loc	Shelter	Yes	>evac time	collect data	--	--	

14	Work/Home/ Other loc	Shelter	No	>evac time	collect data	--	--		
15	Work/Home/ Other loc	Virtual shelter	Yes	>evac time	generate a new trip	home	current time	Number of en route vehicle -1	
16	Work/Home/ Other loc	Virtual shelter	No	>evac time	generate a new trip	nearest shelter	current time	Number of en route vehicle -1	

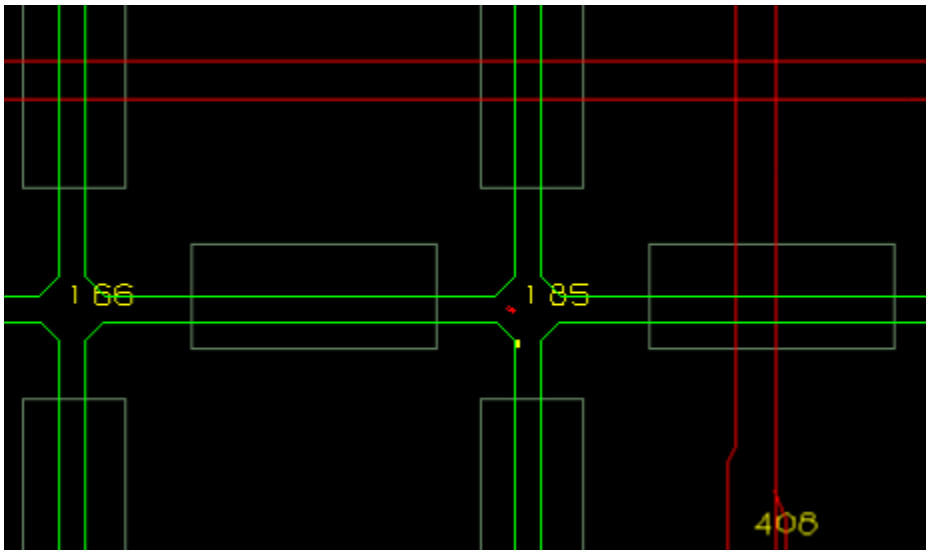


Figure 3-8 (a). Arrived vehicle – before reach its destination

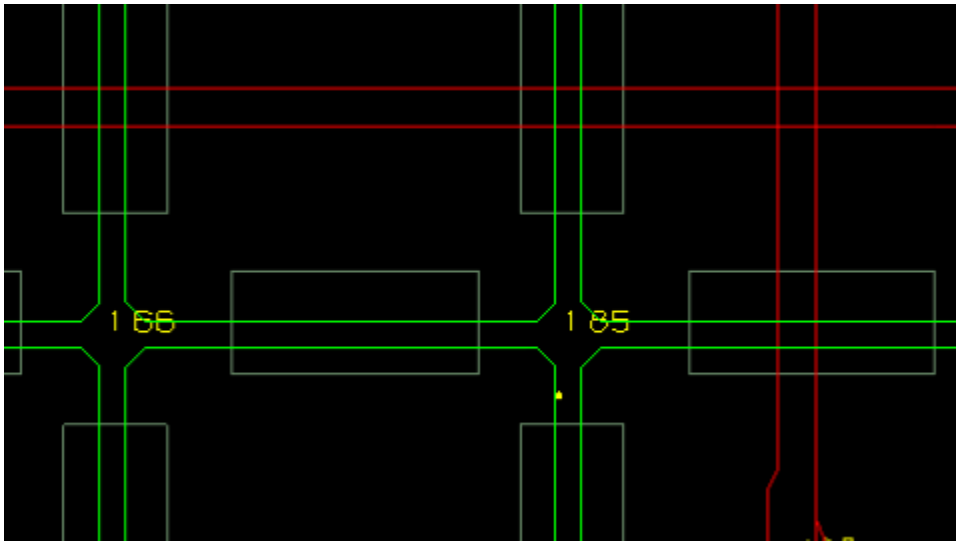


Figure 3-8 (b). Arrived vehicle – vehicle disappeared when reaches its destination

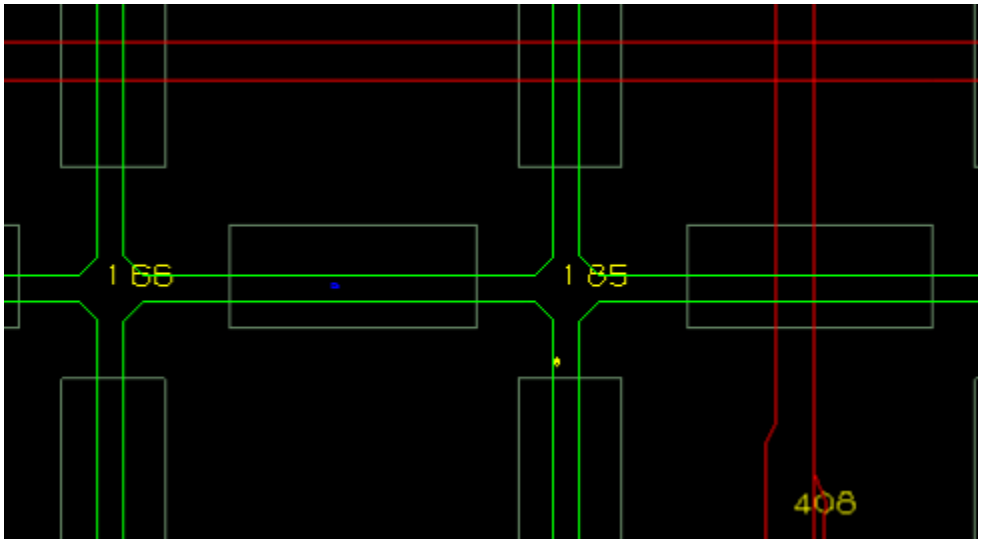


Figure 3-8 (c). Arrived vehicle – new vehicle generated

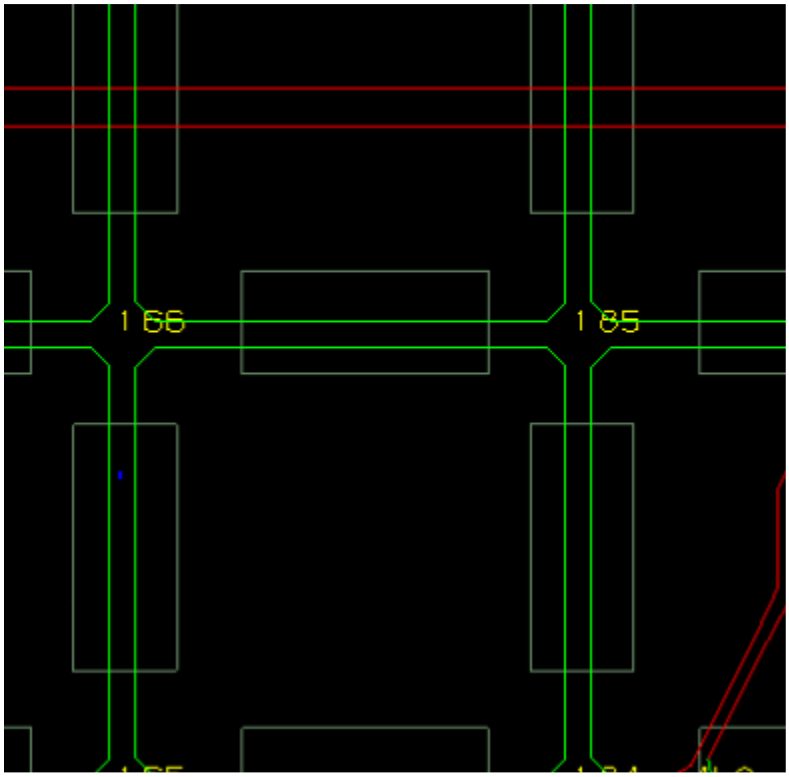


Figure 3-8 (d). Arrived vehicle – new chained trip

3.9 Sub-Model of Consolidated Household Vehicles

One of the key features of the developed API tool is to model the behavior observed by social scientist that many drivers consolidate and evacuate as a social unit. To model such behavior, a vehicle that belongs to a consolidating family will return to its home first when evacuation begins, while a non-consolidating vehicle will evacuate to the closest shelter. After all vehicles associated with a household arrive at their home, they then evacuate together in a single vehicle. In this sub-model, we study the drivers' consolidated household evacuation behaviors at a microscopic level. Based on our knowledge, there are no existing simulation tools that are capable of simulating this kind of behaviors.

The most challenging part in simulating such consolidated household evacuation behaviors is to generate a new consolidated trip because no one else can do this. Similar to the procedure proposed in the previous section, a set of dummy vehicle is used to for this purpose. This type of dummy vehicle is also generated during the data initialization process with a very large departure time to ensure that this vehicle will not be sorted and released. Therefore, we have two different types of dummy vehicles. One is used for creating the new chained trips during the simulation, and the other is used for generating a consolidating trip in this sub-model. To show the difference between these two different types of dummy vehicles, at the initialization, the large departure time for the consolidating dummy vehicle is $t = 1{,}000{,}000$ seconds, while for the new trip dummy vehicle is $t = 2{,}000{,}000$ seconds

One of the significant features of this study is to employ a dynamic waiting time when generating the new consolidated trip. As mentioned above, traditional studies, which are capable of modeling vehicle's trip-chain activities, adopt deterministic (i.e. pre-defined) and static waiting time to decide generation of the consecutive trip. In this proposed tool, the dynamic waiting time is applied to imitate the time duration that the evacuees spend at home for packing essentials and valuable items before hit on the road, as shown in the figure 3-9.

Another significant improvement of this tool is to model the preparedness time of the evacuees for the evacuation. For instance, for a family that has more than one independent family member, some members who arrived at home earlier may find out the evacuation and start packing items while waiting for other members to arrive. In this way, once the last family member returns to home, the whole family can leave home for safety as soon as possible. To model it, in this sub-model, the awareness time at which the evacuee is aware of the evacuation is taken into consideration. Therefore, the departure time of the new consolidated trip must be the maximum time between the last family member's arrival time and the earliest time at which someone in the family was aware of the evacuation, plus a random delay time that is used for packing stuffs.

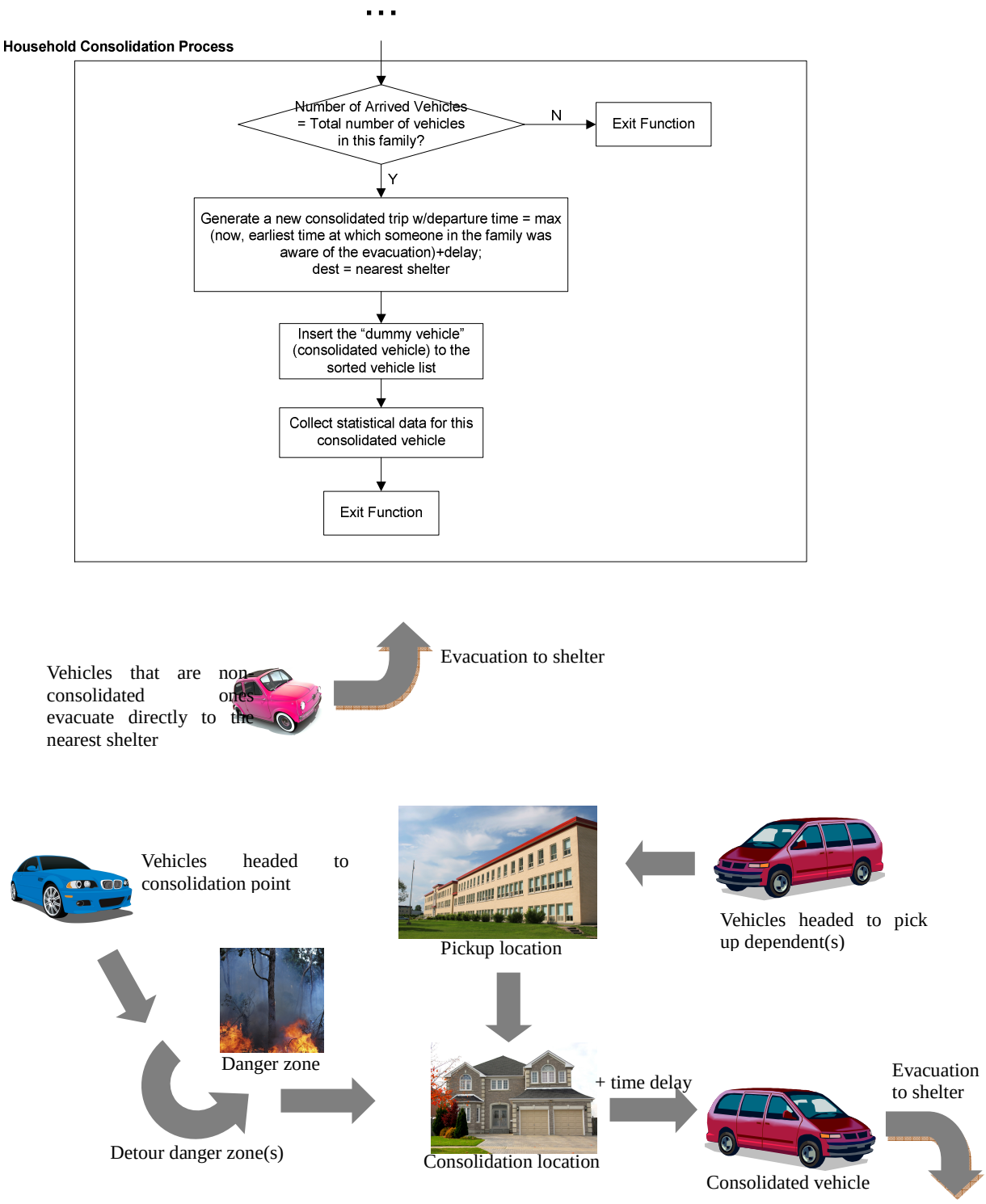


Figure 3-13. Proposed Vehicle Consolidation Procedure

After that, we insert the dummy vehicle that carries the information of the new consolidated trip back to the sorted vehicle list, which is a challenging process. This kind of insertion process is described as follows, which is also the same process as the one implemented in the generation of chained trips in the sub-model of arrived vehicles (as discussed in section 3.8). Because of the space limits, we have these two similar processes to be discussed together as follows.

Step 1: We compare the dummy vehicle's departure time with the **last** vehicle's one in the sorted vehicle list. If the dummy vehicle's is greater, we insert the dummy vehicle at the **end** of the vehicle list. Otherwise, we insert the dummy vehicle in the proper location of the vehicle list. Figure 3-15 shows an example.

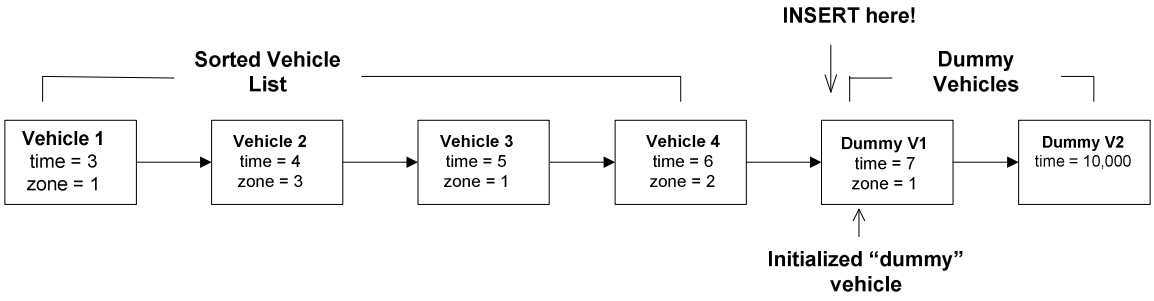


Figure 3-14. Insert the "dummy vehicle" at the end of the sorted vehicle list

Step 2: Except for the departure time, we compare the dummy vehicle's current location (i.e. zone index) with the **last** vehicle's one. If the dummy vehicle's zone index is greater than the last node's, we insert the dummy vehicle at the **end** of the vehicle list. Otherwise, we insert the dummy vehicle in the proper location of the vehicle list, as shown in Figure 3-16.

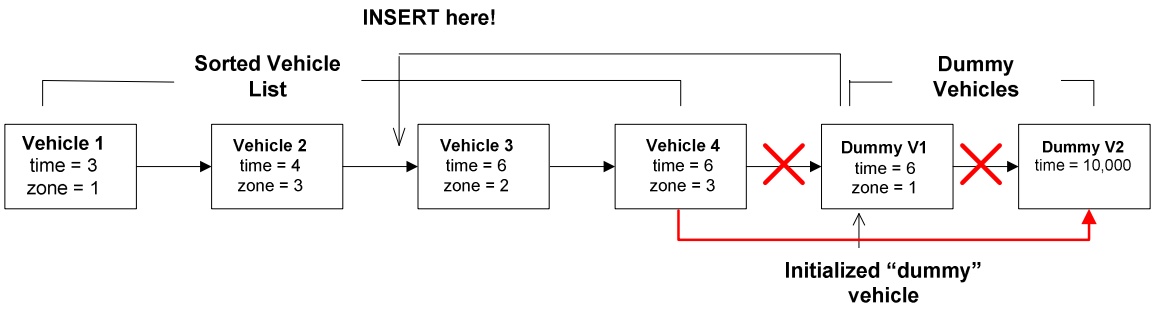


Figure 3-15. Insert the "dummy vehicle" at the end of the sorted linked list by considering its zone index

Step 3: We compare the dummy vehicle's departure time with the **first** vehicle's departure time in the sorted vehicle list. If the dummy vehicle's is less, we insert the dummy vehicle at the **front** of the vehicle list. Otherwise, we insert the dummy vehicle in the proper location of the vehicle list. Figure 3-17 shows an example.

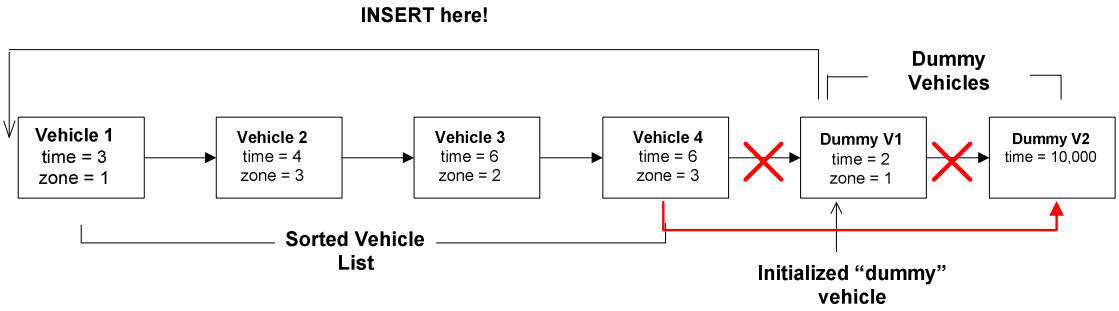


Figure 3-16. Insert the "dummy vehicle" in the front of the sorted vehicle list

Step 4: As similar as shown in step 2, we then compare the dummy vehicle's current location (i.e. zone index) with the **first** vehicle's one. If the dummy vehicle's zone index is less than the first node's, we insert the dummy vehicle at the **front** of the vehicle list. Otherwise, we insert the dummy vehicle in the proper location of the vehicle list, as shown in Figure 3-18.

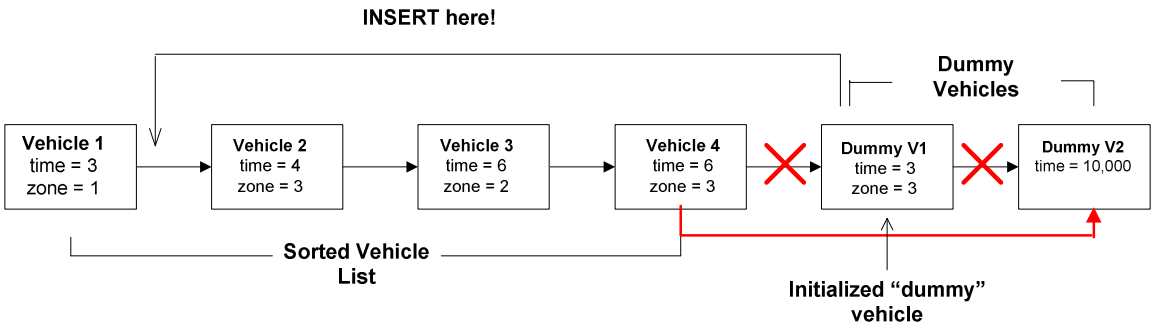


Figure 3-17. Insert the "dummy vehicle" in the front of the sorted linked list by considering its zone index

This sub-model is implemented to augment the core Paramics simulator by using the same API function *qpx_VHC_arrive()* as we used in the previous section. The flowchart that we used to model consolidated household evacuees' behaviors is displayed in the Figure 3-9.

Next a group of figures demonstrates the simulation of the consolidation approach after the implementation.

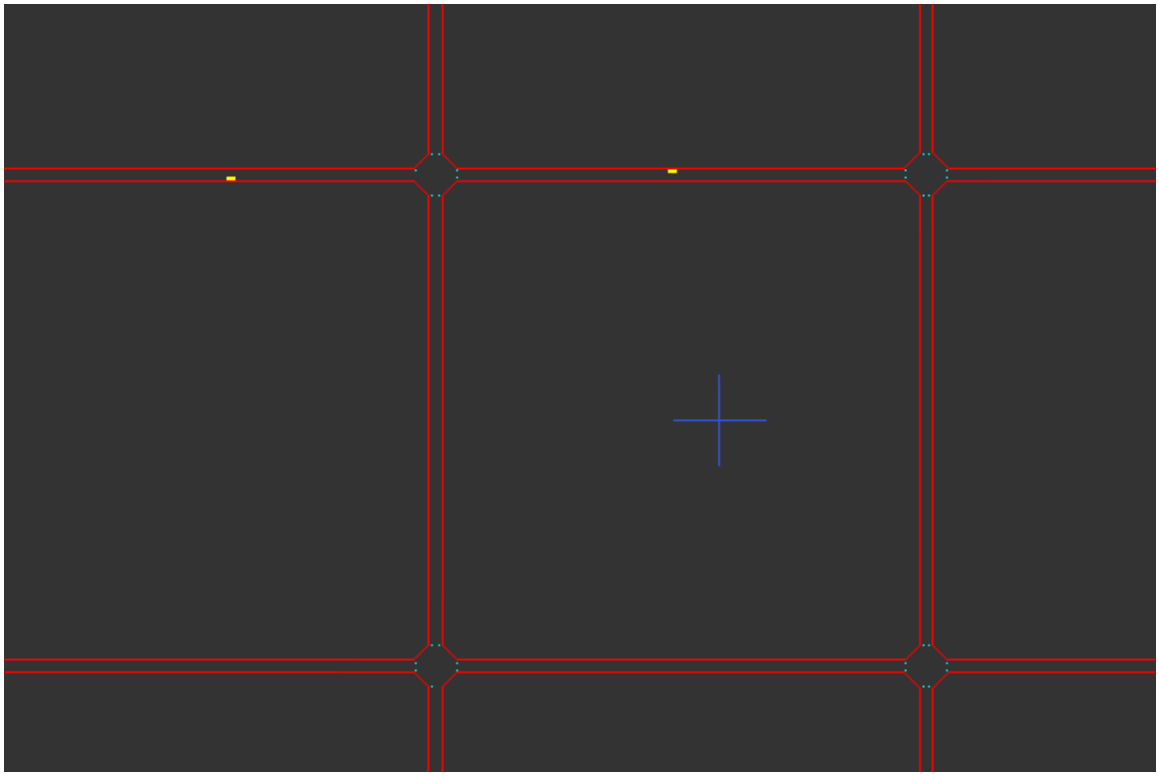


Figure 3-15 (a) Household consolidation – Two vehicles moved towards their home

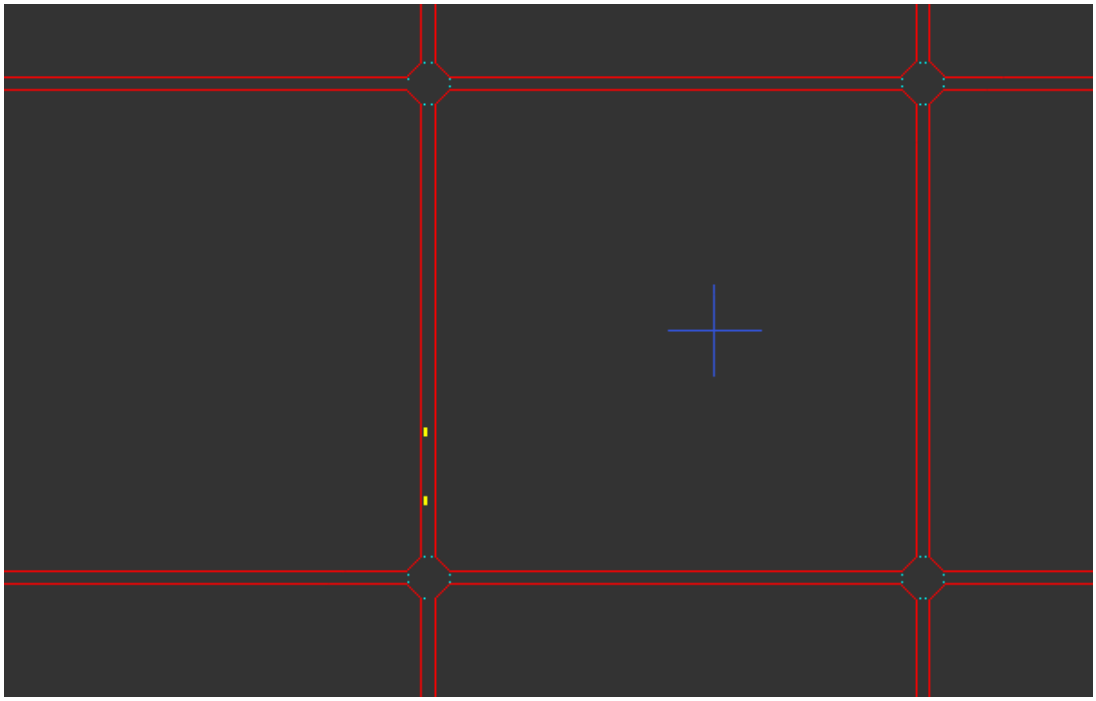


Figure 3-15 (b) Household consolidation – Two vehicles moved towards their home

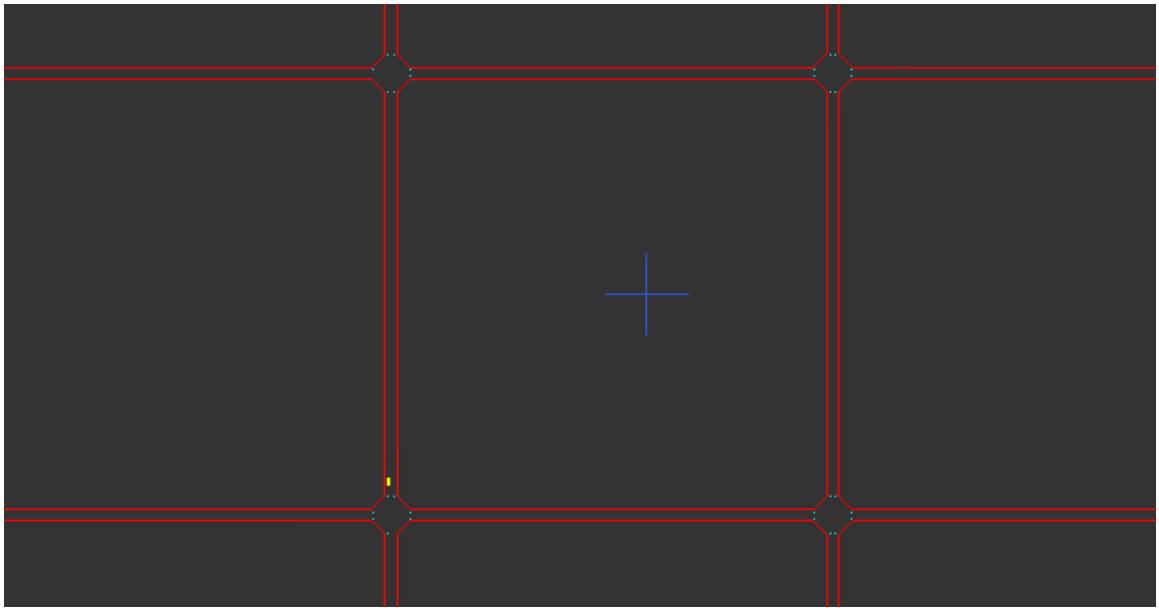


Figure 3-15 (c) Household consolidation – First vehicle disappeared at its home

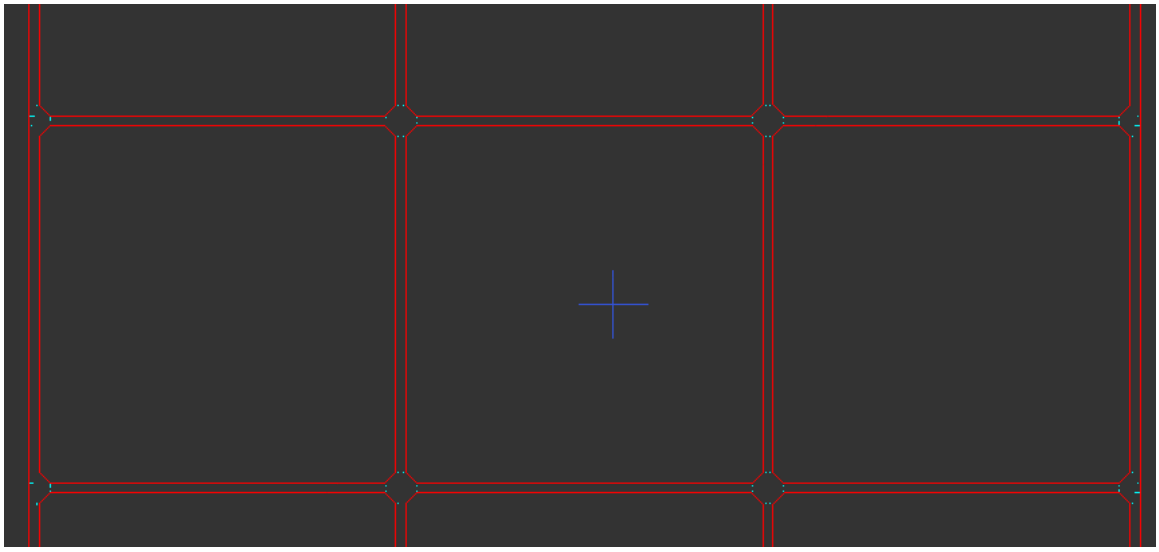


Figure 3-15 (d) Household consolidation – another vehicle also disappeared at its home

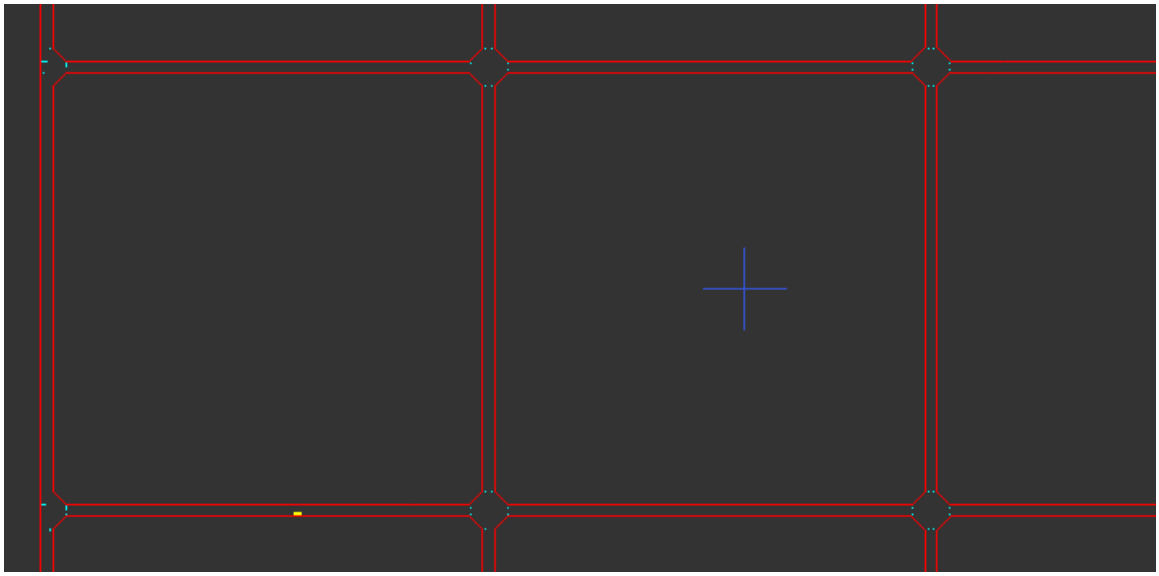


Figure 3-15 (e): a new consolidated trip is generated

Chapter 4: Household Consolidation during Evacuation

In this chapter, a general network model is developed to evaluate the evacuation performance varying under the influence of evacuees' household social behavior. The tool developed in Chapter 3 is implemented. It also examines the interactions between the evacuation consolidation behaviors and traffic conditions, in low demands and high demands conditions.

4.1 Consolidation by Household

A general road network model is designed as a test bed for this study. To simplify our study, the road network is developed with homogeneous parameters, such as the same length and width for each block. No traffic signal control was implemented in this network so that the results of the simulated evacuation study can be attributed directly to the differences between evacuation directly via shortest path and evacuation after consolidating as a family unit.

The simplified road network is coded into Paramics as shown in Figure 4-1. Roads in the simulated network are set as urban roads as shown in green. Each link length is about 0.5 mile so the total network is 10 miles X 10 miles. Each road segment has two lanes, one in each direction, with the same speed limit of 30 mph. Two freeway rings and two crossing freeways are shown in red, with speed limits of 60 mph.

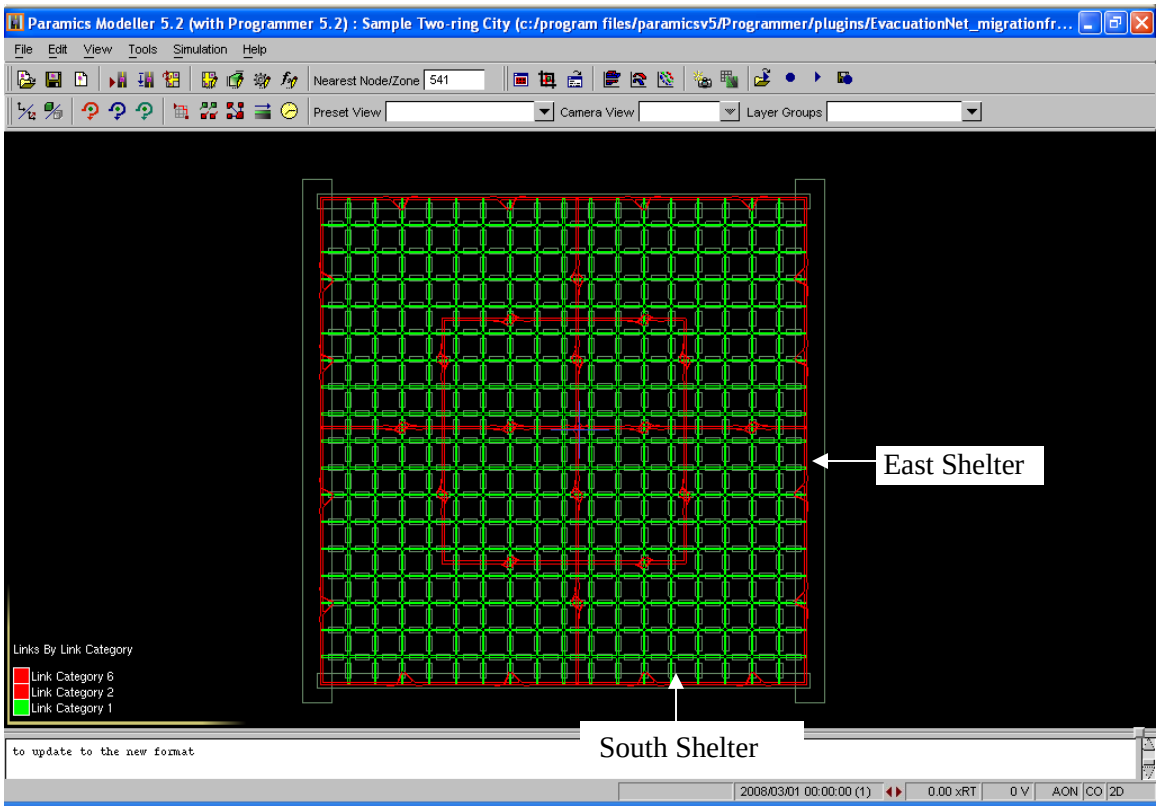


Figure 4-1. Simulated road network

Each link is designated as a zone, as Paramics' origin-destination matrix for individual vehicles uses the unit of zone. In this way, a vehicle can be controlled to move from one origin zone, a link, to a destination link by defining the origin zone and destination zone. Finer division of the zones for network-wide evacuation is not necessary from the system perspective of view. The simulated network has a total of 542 nodes, 1677 links, 33 interchanges, and 616 zones. The two boundary zones, as shown in Figure 4-1, represent two shelter locations in the east and south of the network. For most coastal cities, in an emergency event, such as a hurricane, evacuation is typified with evacuating in a signle direction or two directions.. Chapter 7 will investigate the

influences of different numbers of shelters in the network on the household consolidation evacuation.

The network is seeded with 1,000 families and 10,000 vehicles, with an assumed bivariate normal distribution about the geographic center of the network. For the distribution of departure time of the vehicles, a triangular distribution is assumed. For the simulated scenarios, normal driving tactics are assumed.

First, we present the proposed API tool by identifying its capability of modeling consolidation by household. To illustrate the effect of household consolidation on evacuation efficiency, a range of scenarios are compared:

- All vehicles take the shortest route to safety.

This is the typical assumption used in traditional evacuation studies. Every vehicle would seek the shortest or quickest route to safety. Each vehicle is assigned a link on the boundary of the network closest to its current location. While not considered a very realistic scenario, it is a reasonable approximation to the network capacity to evacuate. It also provides a baseline that is similar to other reported results, against which to compare the scenarios that consider some household consolidation.

- All families consolidate before evacuation.

All family units with two or more members will consolidate, and then evacuate as a unit. The home is assumed to be the de facto meeting location. Once all vehicles in the family's fleet arrive, a new consolidated trip is generated.

- Some fractions of families consolidate before evacuation.

Families with no dependents that have evacuation plans may not need to meet at home, and if communication channels are still available, some families may decide to meet at a destination shelter instead of meeting at home. Some emergency situations might be so life-threatening that some of the family members will not consider consolidating before evacuation. For a better understanding, consolidation ratios of 10% increment, i.e. 10%, 20%, …, 80%, and 90%, are shown, although any specific ratio can be implemented in the tool.

The simulations uses half-second time intervals. Evacuation is assumed to start after a warmup time, i.e. at t = 15 minutes, which is an initialization time period for filling vehicles into the network. Evacuees are assumed to be aware of the evacuation with a random delay time as the evacuation starts. For those household vehicles who return to their homes first, before they evacuate together as a single unit, a dynamic delay time for the family to pack essentials and valuable items is assumed to vary uniformly between 0 and 30 minutes.

The elapsed time for a percentage of residents to arrive at the boundary link, and the percentage population to be evacuated, are used as metrics to evaluate the evacuation performance.

Figure 4-2 shows the traffic flow performance under 0% consolidation, 50% consolidation, and 100% consolidation scenarios. The blue line represents the performance of the 0% consolidation scenario, the yellow line the 50% consolidation scenario, and the pink line the 100% consolidation scenario. As expected, the scenario with all families consolidating before the evacuation takes longer to fully evacuate than the 0% consolidation scenario. An interesting thing is that the 50% consolidation and 100% consolidation curves end at the same time, which means to evacuate 100% of the population, the model suggests it will take the same time for the 50% consolidation scenario and the 100% consolidation scenario.

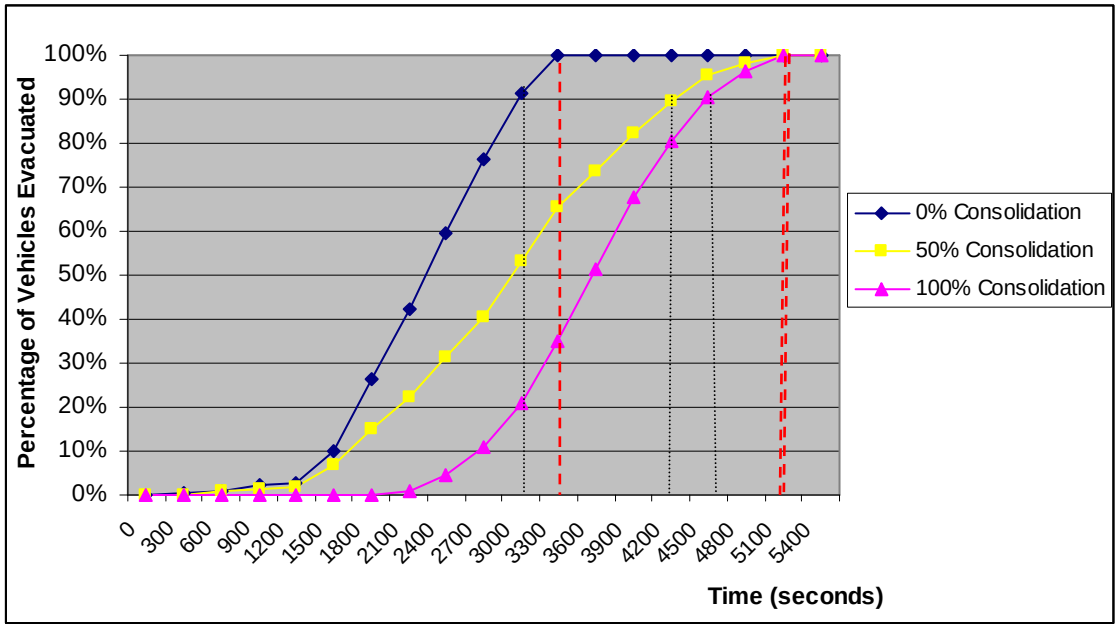


Figure 4-2. Performance of consolidation by household under three scenarios with low demand

Furthermore, Figure 4-3 shows the consolidation ratios with 10% increments. This figure shows that the times for evacuating 100% of the population from the 10% consolidation scenario to the 100% consolidation scenario are almost identical. However, for evacuating 90% of evacuees, the 10% consolidation scenario takes about 3200 seconds, the 30% consolidation scenario takes about 3900 seconds, the 50% consolidation scenario takes about 4250 seconds, and the 100% consolidation scenario takes about 4500 seconds, which lead to significantly different performance. Many previous studies (Sheffi et al. 1982, KLD 1984, Rontiris and Crous 2000, Tuydes and Ziliaskopoulos, 2004, etc.) used the time to evacuate all evacuees outside of an evacuation area as one of the primary measures of effectiveness (MOEs) for evacuation. The reason is that it is easy to talk about 100% evacuees being evacuated. However, in reality, people do not use the 100% since some persons may not leave the area at the end of the evacuation. A more common metric is to take 90% of evacuees out of the network. As shown in Figure 4-2 and 4-3, it does make a difference. Therefore, we use 90% as a metric as suggested by the simulation.

Figure 4-3 also reveals that, as the consolidation rates go from 0% to 50%, the differ more in their upper portions, which represent the last half of the evacuees. When the consolidation rates are higher, from 60% to 100%, the lower portions of the curves differ the most, corresponding to the first half of the evacuees.

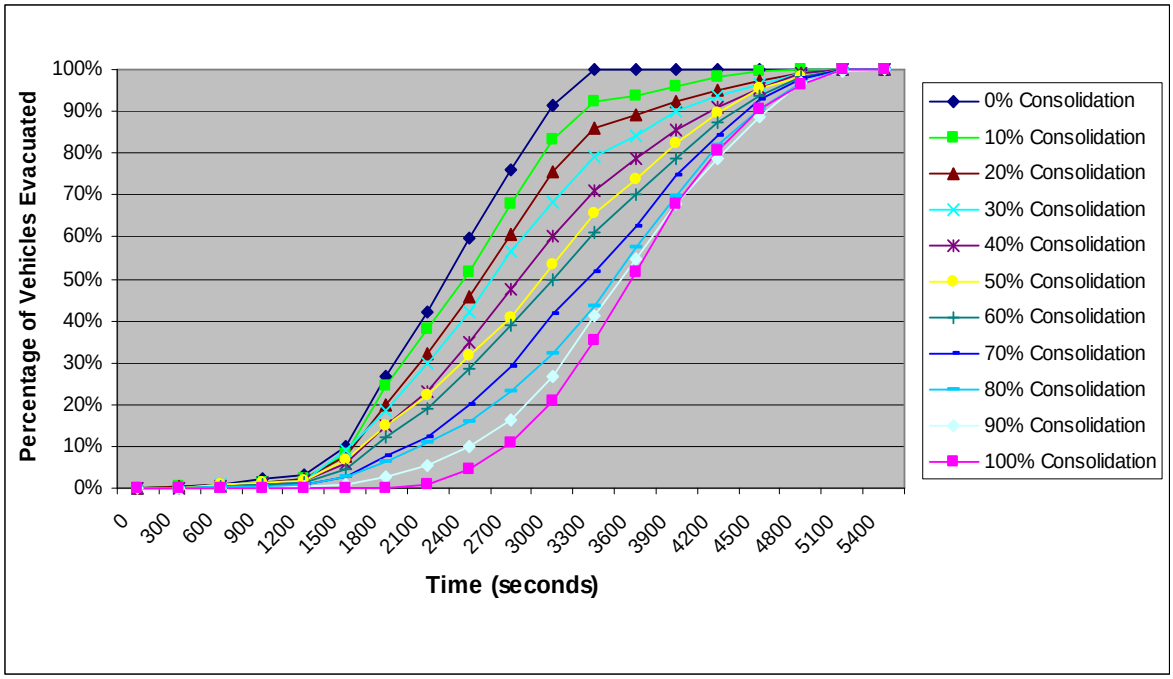


Figure 4-3. Performance of consolidation by household under various scenarios with low demand

Furthermore, as documented in Table 4-1, the evacuation time for 90% of vehicles for the 0% consolidation scenario is about 3000 seconds, and for the 100% consolidation scenario it is about 4500 seconds, which is 50% longer than 0% consolidation. This result agrees strongly with the results of 50% longer if evacuation is preceded by the members meeting at home, as obtained by Murray-Tuite and Mahmassani (2003, 2004). Besides, our results also show that, with half of the families consolidating as a unit for evacuation, it takes about 4250 seconds, which is approximately 42% longer to evacuate 90% of the families compared to no families consolidating.

Table 4-1. Network clearance time of various scenarios with low traffic volume situation

Time (sec)	100%	90%	80%	70%	60%	50%	40%	30%	20%	10%	0%
0% Consolidation	3240	2960	2760	2600	2400	2250	2040	1860	1700	1500	0
100% Consolidation	5200	4500	4200	3950	3750	3560	3420	3200	3000	2660	0
50% Consolidation	5200	4250	3850	3500	3120	2950	2700	2350	2000	1650	0

Figure 4-4 shows the trend from a different perspective. It shows that the yellow bars, which come from the 50% consolidation scenario, are close to the blue bars, which are for the 0% consolidation scenario, at the right end of the figure when evacuating 10% to 20% of the people. The main reason is that at the beginning of the evacuation, there are more non-consolidated vehicles that take the shortest path to the safety, so those are the same group of vehicles to be evacuated for both the 0% and 50% consolidation scenarios. Later on, at the left end of Figure 4-4, to evacuate 90% to 100% people, the yellow curve gets closer to the pink one, which is for the 100% consolidation scenario. It shows that there are more and more families who finished their consolidation process entering the network at the final stage of the evacuation. Because the yellow curve is not increasing linearly from the blue curve to the red one, it would be unwise to simply interpolate between the two extreme scenarios.

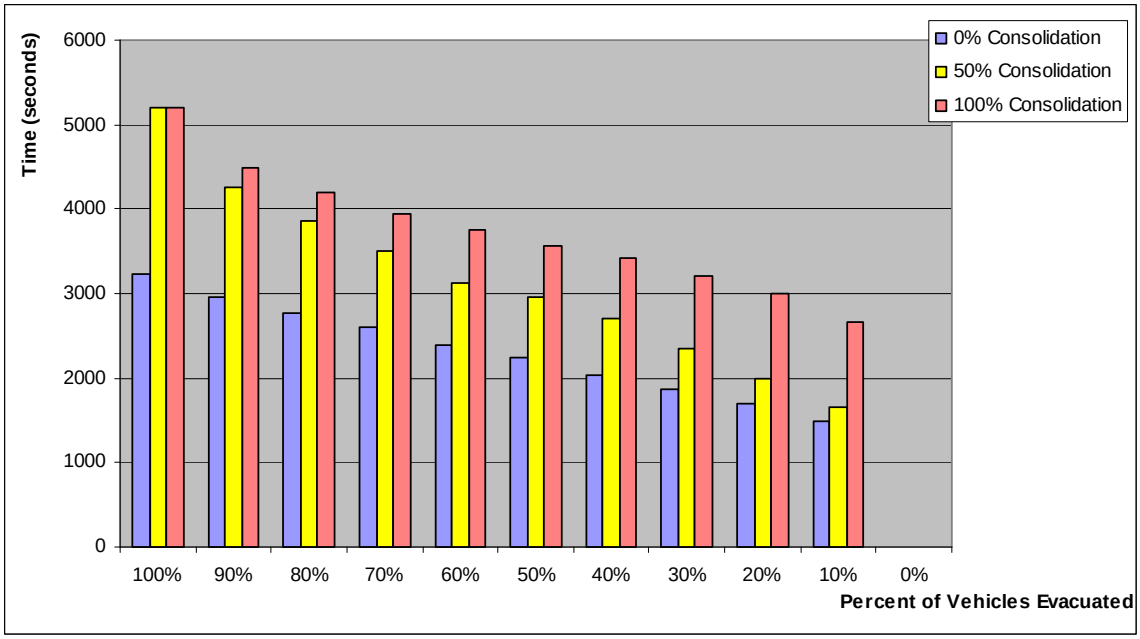


Figure 4-4. Comparison of arrival time of various scenarios with low demand situation

The study of consolidation by household evacuation shows a significant difference in evacuation time compared to the assumption of all evacuees taking the shortest route from the danger. In this study, the inbound flow generated by the 100% consolidation scenario delay the process of the traffic leaving the network. In contrast, in the 0% consolidation scenario, all vehicles leave the network immediately from the start of the evacuation. Therefore, it takes longer for the 100% consolidation scenario to fully evacuate than the 0% consolidation scenario. However, it also raise the question: once the traffic flows become heavy in the studied network, if all vehicles take the shortest route to safety under the 0% consolidation scenario, is there a chance that the network will be congested by these vehicles so that no one could move? If this were true, the scenario in which all vehicles leave immediately may take longer to fully evacuate than if

all families consolidated before the evacuation. This scenario will be investigated in the next section.

4.2 Consolidation Pattern with Heavy Demand

After the evacuation begins, there is a rapidly rising demand in the network, especially for emergency events that have no early warnings. This gives rise to the question: what will happen to the traffic patterns that incorporate household consolidation under such heavy demand situation?

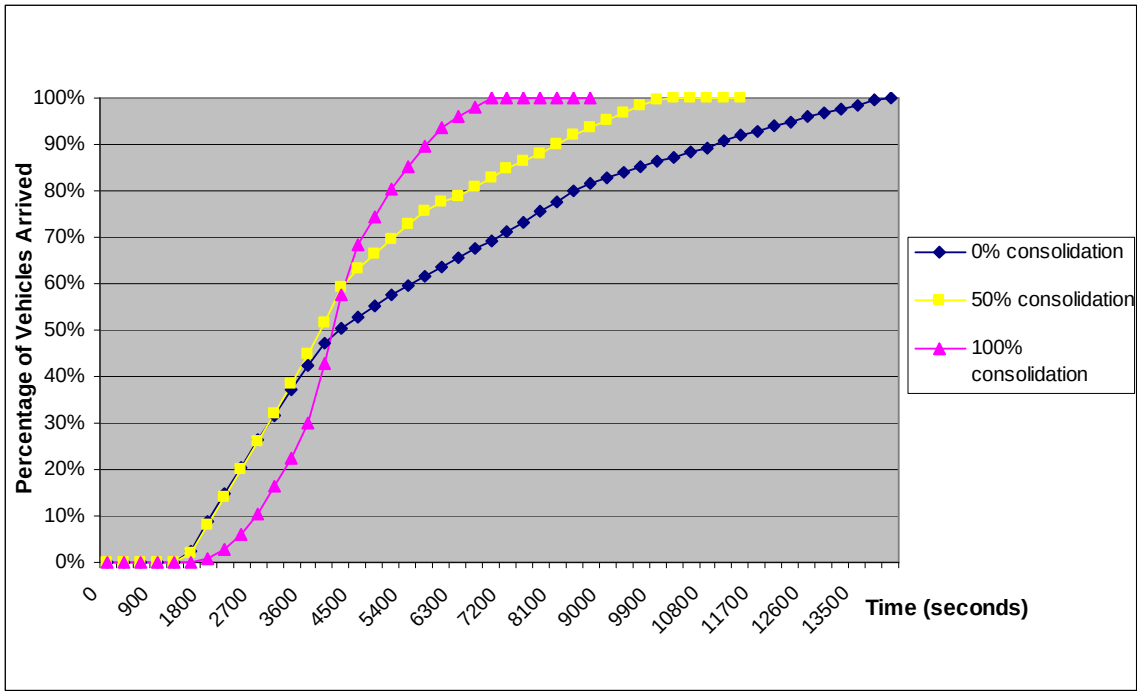
In this section, we investigate this problem by conducting a case study of how the traffic pattern changes with different household consolidation behaviors under heavy network demands. We also compare the results obtained from this study to those from the previous section.

The simulation test network used for this case study is the same as the one we described in section 4.1. In addition, in this case, the network is seeded with 5,000 families and 100,000 vehicles, which represent a heavy demand situation. There is an initialization time period t = 20 minutes in the simulation, which allows more vehicles to fill into the network.

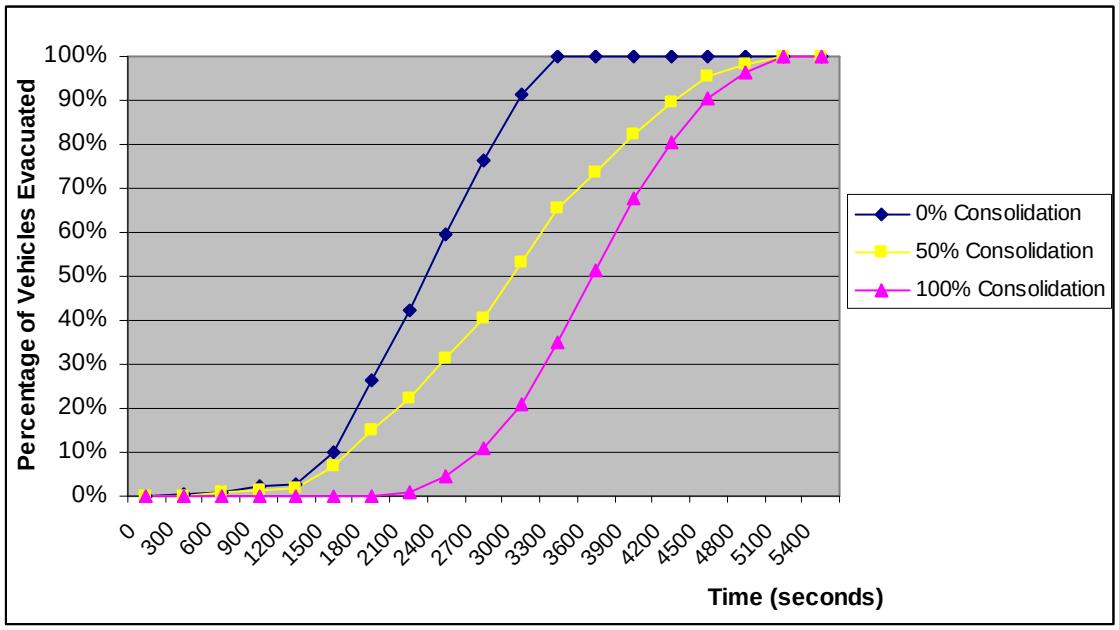
Next, we conduct the case study by identifying the effects of the heavy demands on the following three scenarios. In this case, 0% consolidation scenario has all vehicles take the shortest route to safety, 100% consolidation scenario has all families consolidate before the evacuation, and 50% consolidation scenario has fifty percent of families consolidate before the evacuation.

The simulation results are summarized in figure 4-5. It shows that, in contrast with those results from the consolidation by household scenarios under low demands in section 4.1, in this case, with the heavy demands, 0% consolidation scenario will take the longest time to fully evacuate compared to 50% consolidation and 100% consolidation scenarios. This is an important result, because it suggests that it may actually be detrimental to have a very high percentage of vehicles attempting an outbound movement simultaneously, because they over-congest the network. This is not necessarily a controllable outcome – people will presumably return home and possibly consolidate regardless of instructions from the government – but it does shed considerable light on what the most useful and efficient traffic management strategies for evacuations might be, like staged evacuation.

A confounding factor, however, is the fact that the number of vehicles departing the network is different under the scenarios. With more consolidation, the ultimate number of vehicles attempting to depart the network decreases, which suggests decreasing amounts of congestion at the exits.



(a) High Demand

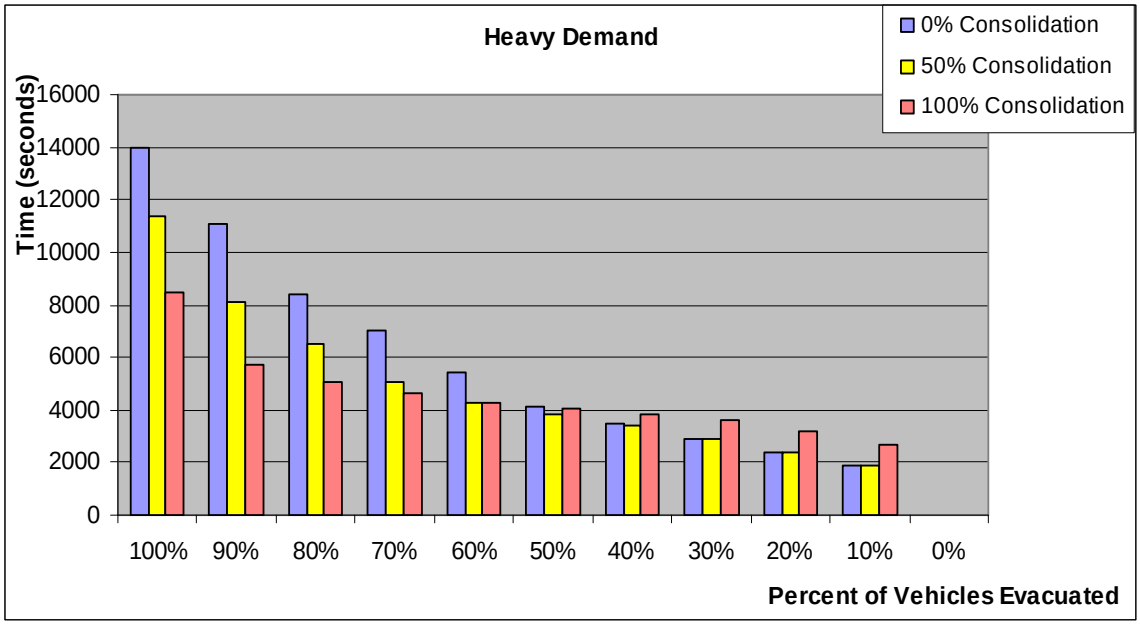


(b) Low Demand

Figure 4-5. Performance of various scenarios on the studied network – heavy demand case vs. low demand case

The differences between heavy and light traffic volumes are demonstrated in Figure 4-6. From these two figures, we observe that, with the low demand, the yellow curve gets closer to the blue curve from 10% to 20% of evacuation, and gets closer to the red curve from 90% to 100% of evacuation. However, with the high demand, when evacuating 10% to 40% of evacuees, the yellow curve is close to the blue curve, and then from 50% to 100%, the yellow curve gets close to the red one. In particular, there are significant differences among the three curves when it comes to evacuate 80% to 100% of population.

The main reason is that, if all vehicles take the shortest route to safety, they congest the network exits. In addition, these vehicles form lines that spill back quickly, and create congestion at other parts of the network. This is a "compounded" effect and means that more people cannot move.



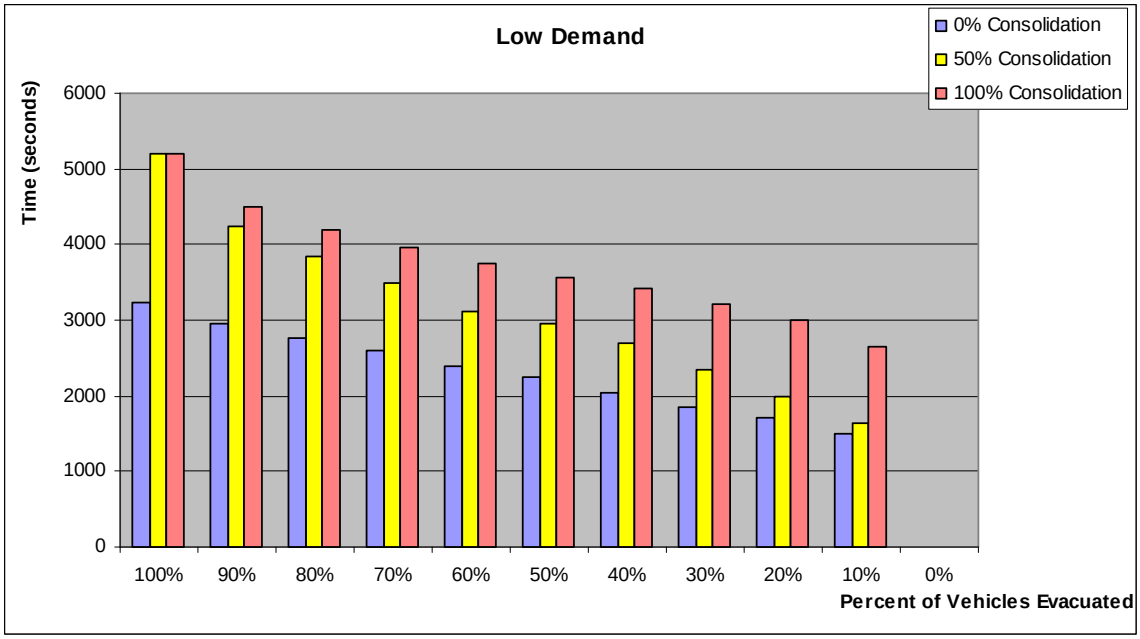


Figure 4-6. Comparison of arrival time of various scenarios with heavy demand vs. low demand situations

Figures 4-7, 4-8, and 4-9 highlight the differences between heavy and light traffic demands under different consolidation scenarios. For example, in Figure 4-7, it indicates that, with 0% consolidation scenario, it takes significantly longer to evacuate 0% to 100% of population with the high demand situation. In Figure 4-8, with 50% consolidation scenario, to evacuate 0% to 40% of evacuees, the high demand case takes less time than the low demand case. However, to evacuate the remaining 50% to 100% of evacuees, the high demand case takes longer than the low demand case. In Figure 4-9, with 100% consolidation scenario, it also takes dramatically longer for evacuating different fractions of evacuees with the high demand situation. Besides, these figures show that low demand has a linear trend, while high demand is quadratic trend. Therefore, it is difficult to estimate the consolidation behavior effects under a high demand situation.

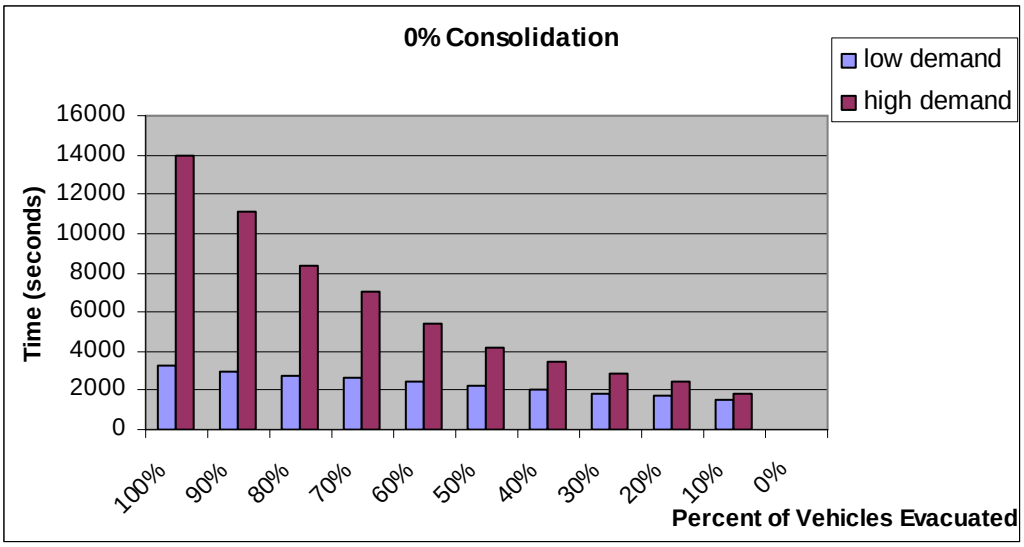


Figure 4-7. Comparison of percent of vehicles evacuated with low vs. high demand under 0% consolidation scenario

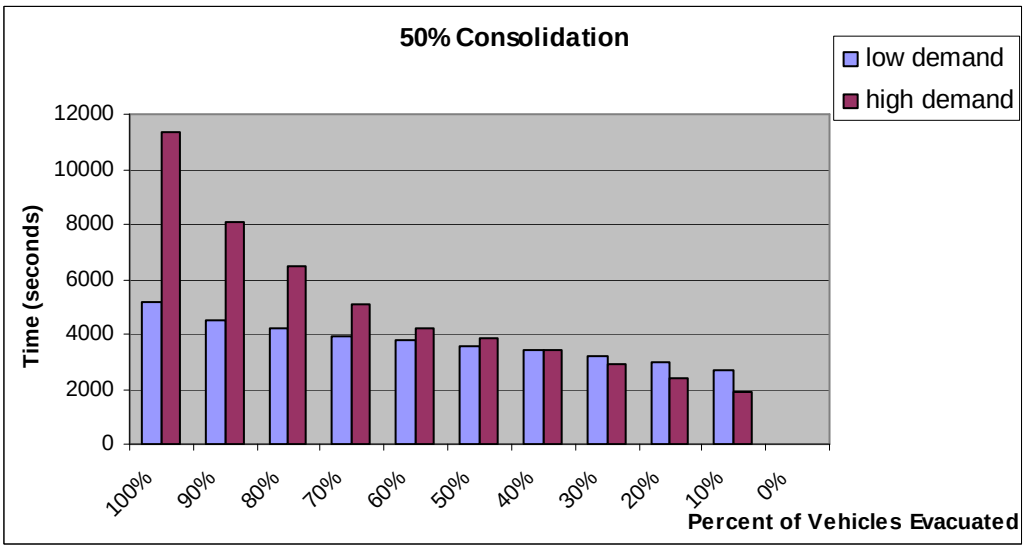


Figure 4-8. Comparison of percent of vehicles evacuated with low vs. high demand under 50% consolidation scenario

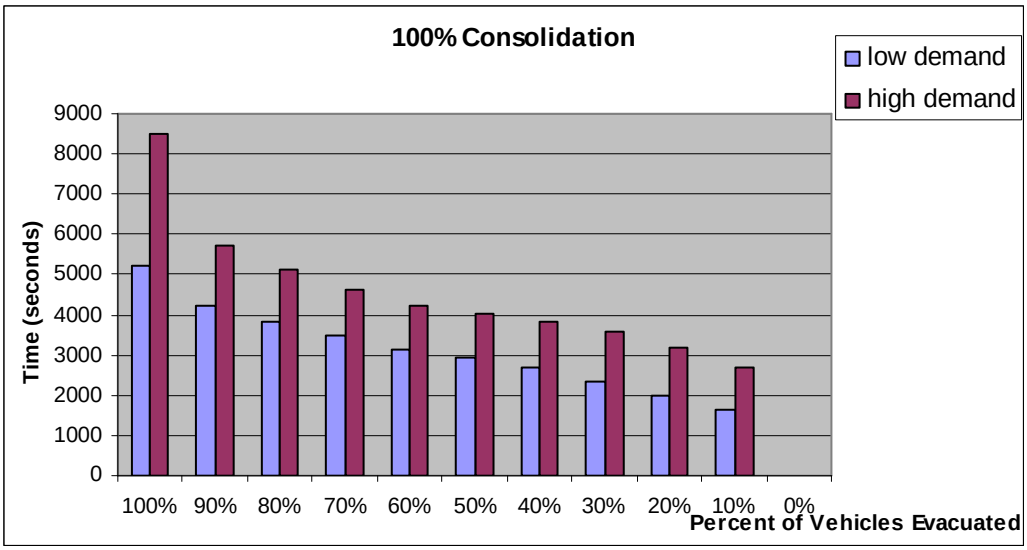


Figure 4-9. Comparison of percent of vehicles evacuated with low vs. high demand under 100% consolidation scenario

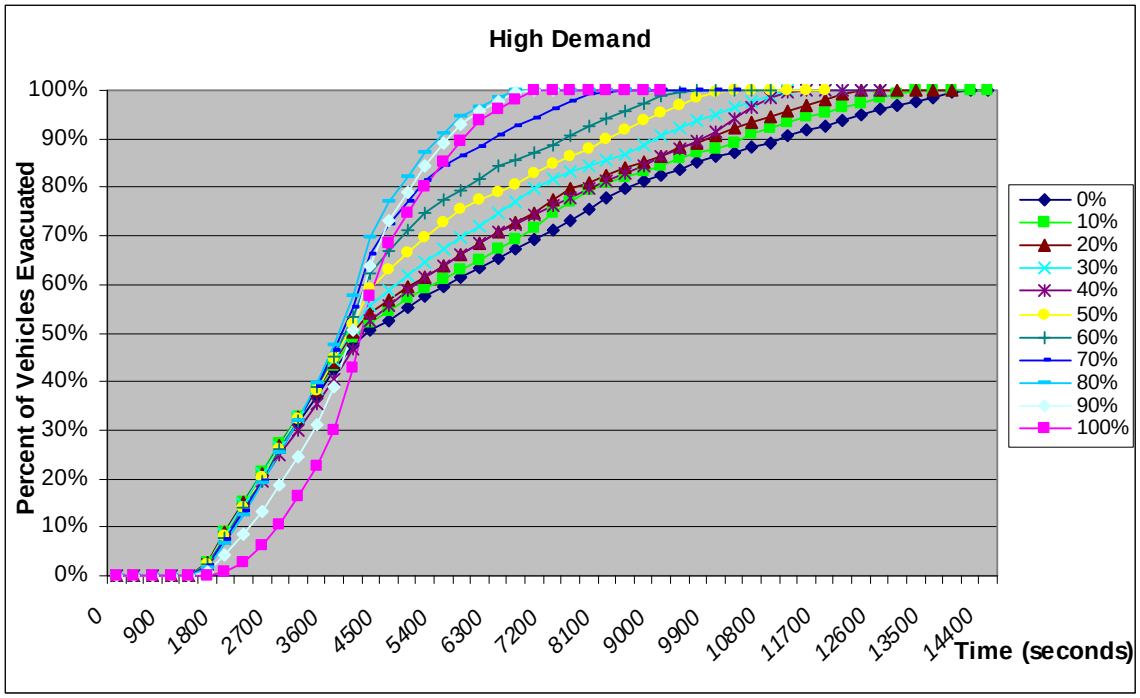
Quantitatively, as shown in Table 4-2 and Figure 4-5, the evacuation time for 90% of vehicles for the non-consolidation scenario is about 11100 seconds, and for the all-consolidation scenario is about 5700 seconds, which is about 50% shorter than the non-consolidation scenario. The results reveal that the heavy demand has a significant influence on the traffic patterns with household consolidation during an emergency evacuation.

Table 4-2. Arrival time of various scenarios with heavy demand situation

Time (Sec)	0% Consolidation	50% Consolidation	100% Consolidation
100% Evacuated	14000	11350	8500
90% Evacuated	11000	8100	5700
80% Evacuated	8400	6500	5100
70% Evacuated	7000	5100	4600
60% Evacuated	5450	4250	4250
50% Evacuated	4150	3850	4050

40% Evacuated	3450	3400	3850
30% Evacuated	2900	2900	3600
20% Evacuated	2400	2400	3200
10% Evacuated	1850	1900	2700
0% Evacuated	0	0	0

Next, we simulate the aforementioned case with 10% increment in the percentage of family consolidation. The simulation results are shown in Figure 4-10, which is compared with the figure from the low demand study. Figure 4-10 reveals that at the beginning, from 0% of families consolidating to 40% of families consolidating, there is no obvious difference between low demand and high demand results. The turning point occurs when the consolidation rate increases from 40% to 50%. The changes from 50% to 100% are more significant than those from 0% to 40%, where each increment pushes the curve to the left side only slightly. These results emphasize the impact of consolidation of household on the traffic patterns with heavy demand situation.



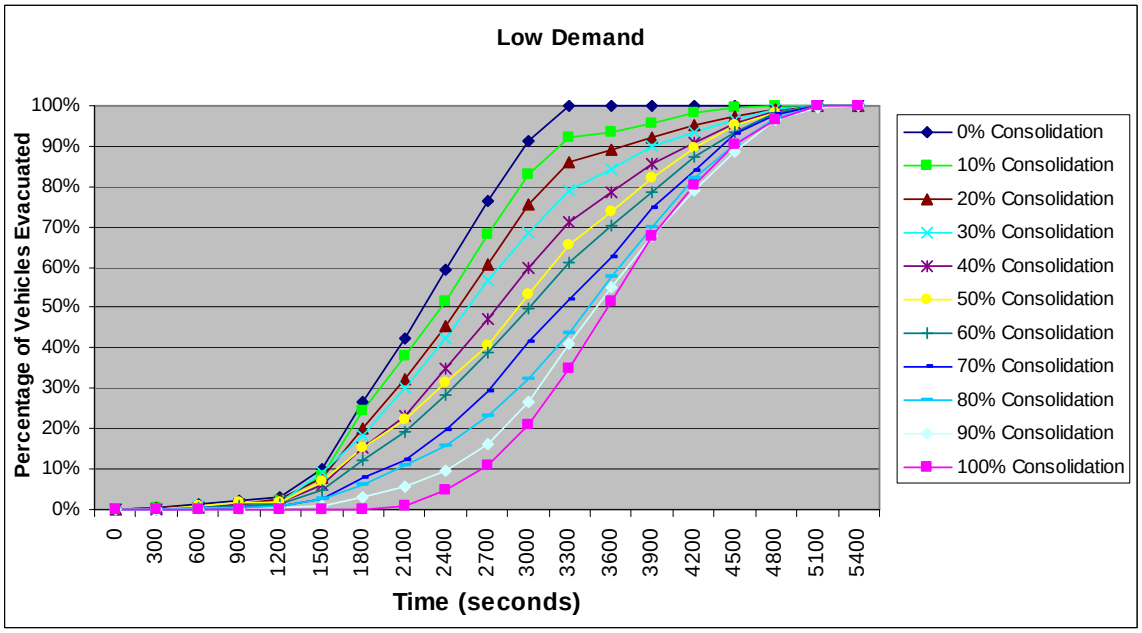


Figure 4-10. Performance of various scenarios on the studied network – heavy demand vs. low demand

It is very important to remember what real effect consolidation has. If we had stuck with Figure 4-10, one might conclude that consolidation improves the evacuation

performance, but the top graph in Figure 4-10 suggests that this is not the case. The reason is that the consolidated vehicles have more people in them. Therefore, in the 100% consolidaton scenario, the number of departing vehicles is less than it would be otherwise. Figure 4-11 shows the total number of vehicles to have arrived at the shelter among different consolidation scenarios. It shows that the 100% line is always behind the 50% line in time, which is in turn always behind the 0% line, as we would expect.

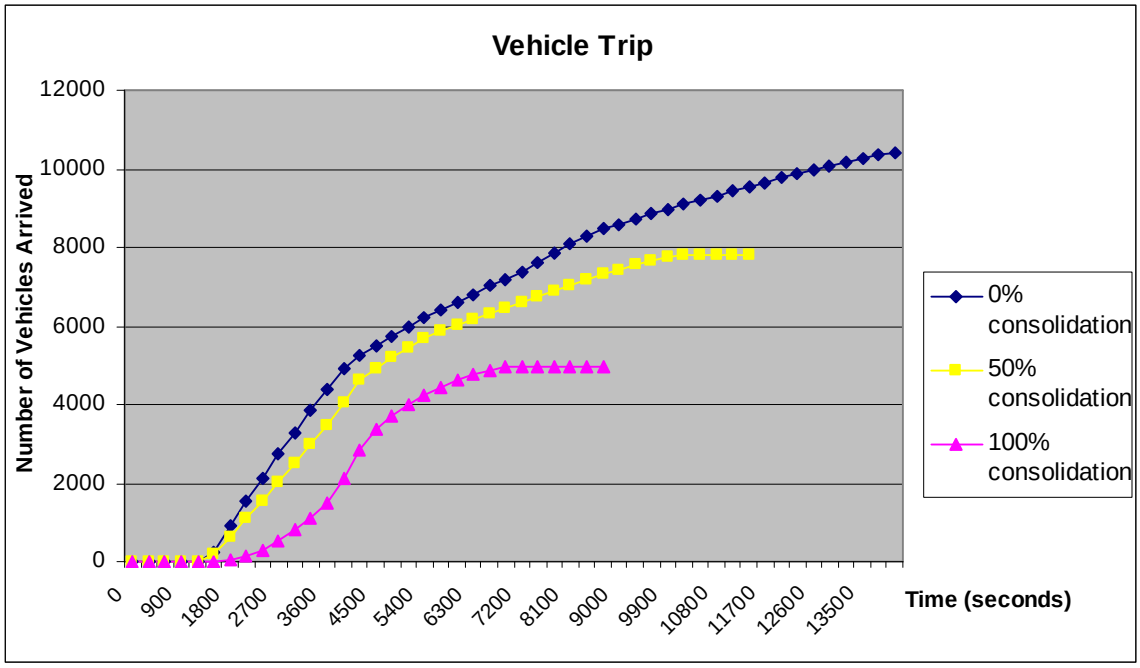


Figure 4-11. Total Number of Vehicles Being Evacuated under Heavy Demand

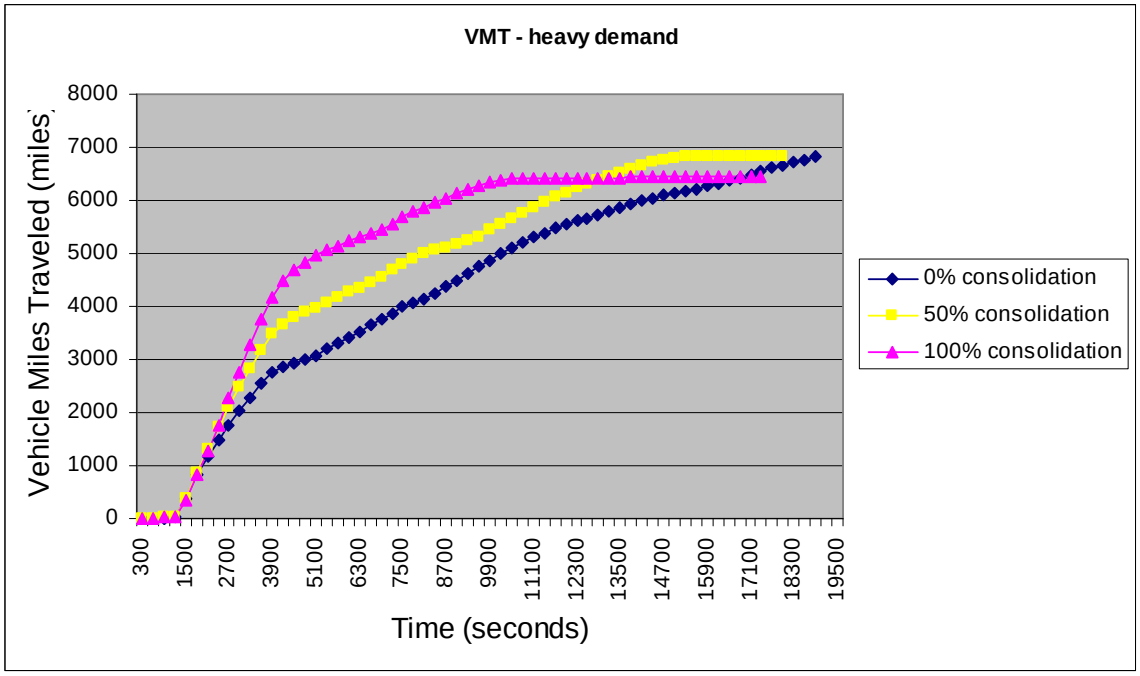


Figure 4-12. Vehicle Miles Traveled under Heavy Demand

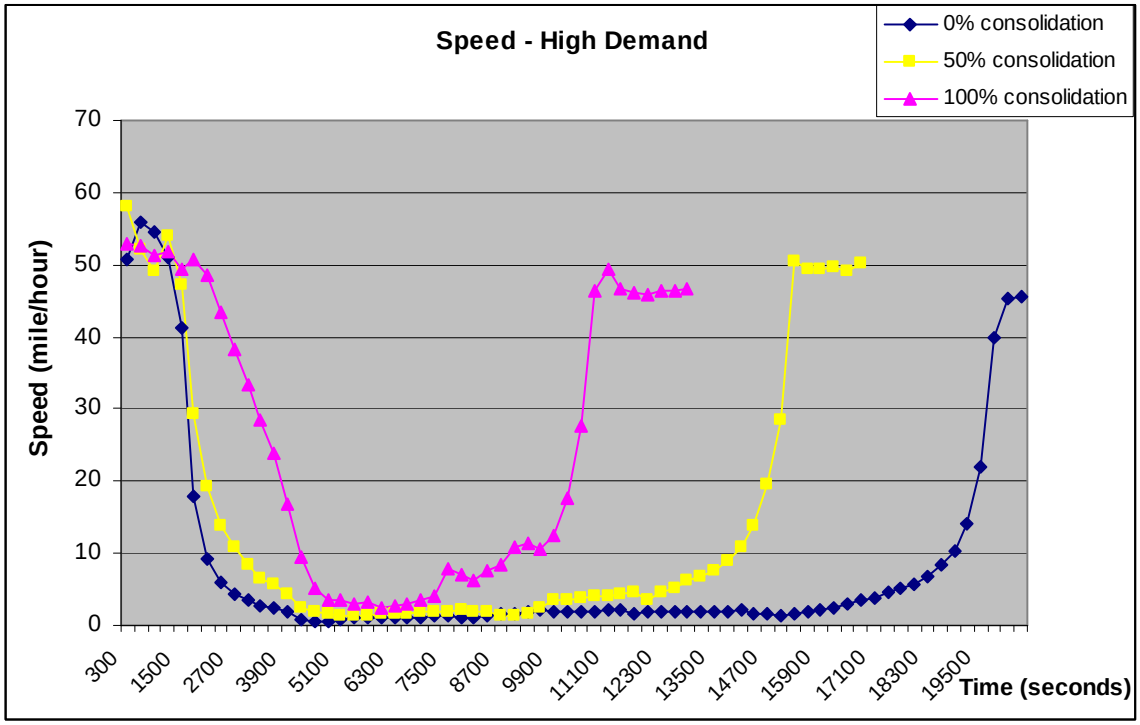


Figure 4-13. Average Speed of Vehicles Being Evacuated under Heavy Demand

Figures 4-12 and 4-13 shows the actual vehicle miles traveled and the average speed for vehicles being evacuated, respectively. It shows that with high consolidation rates, extra vehicle miles traveled are incurred to accomplish the consolidation. The average speed shows the network congestion level. These results validate that with high demands, low consolidation rates seem to produce the longest evacuation times, which means a very high percentage of vehicles attempting an outbound movement simultaneously so that the network becomes over-congested. These figures are also further evidence of the importance of having a good estimate of percent of evacuees likely to consolidate.

Chapter 5: Directional Flow Study

In this chapter, we investigate the traffic volumes entering and leaving the network, as family consolidation is believed to have an impact on such traffic flow patterns. As discussed previously, the study of directional traffic flow plays a critical role in making evacuation management strategies such as the use of contra-flow and signal control operations. However, incorrect assumptions of evacuees' behaviors could lead to ineffective use of these strategies, which may decrease the evacuation rate rather than increase it.

5.1 Overview

In this case study of directional flow, we use the same network as described in the previous section. The network is seeded with 5,000 families and 100,000 vehicles, which represent a heavy demand situation. An evacuation order is assumed to be given after twenty minutes of the simulation. Due to the delays in the dissemination of the evacuation order, evacuees are assumed to be aware of the evacuation with random delays between 0 and 30 minutes, distributed uniformly. For those homebound vehicles who return to their homes first, a delay time that is used for the evacuation preparedness for this family is assumed to vary between zero and half an hour. These numbers are for illustration purposes; any distributions can be incorporated into the API tool. In this study, as shown in Figure 5-1, we assume that there are four shelters, one in each of the four directions of the network.

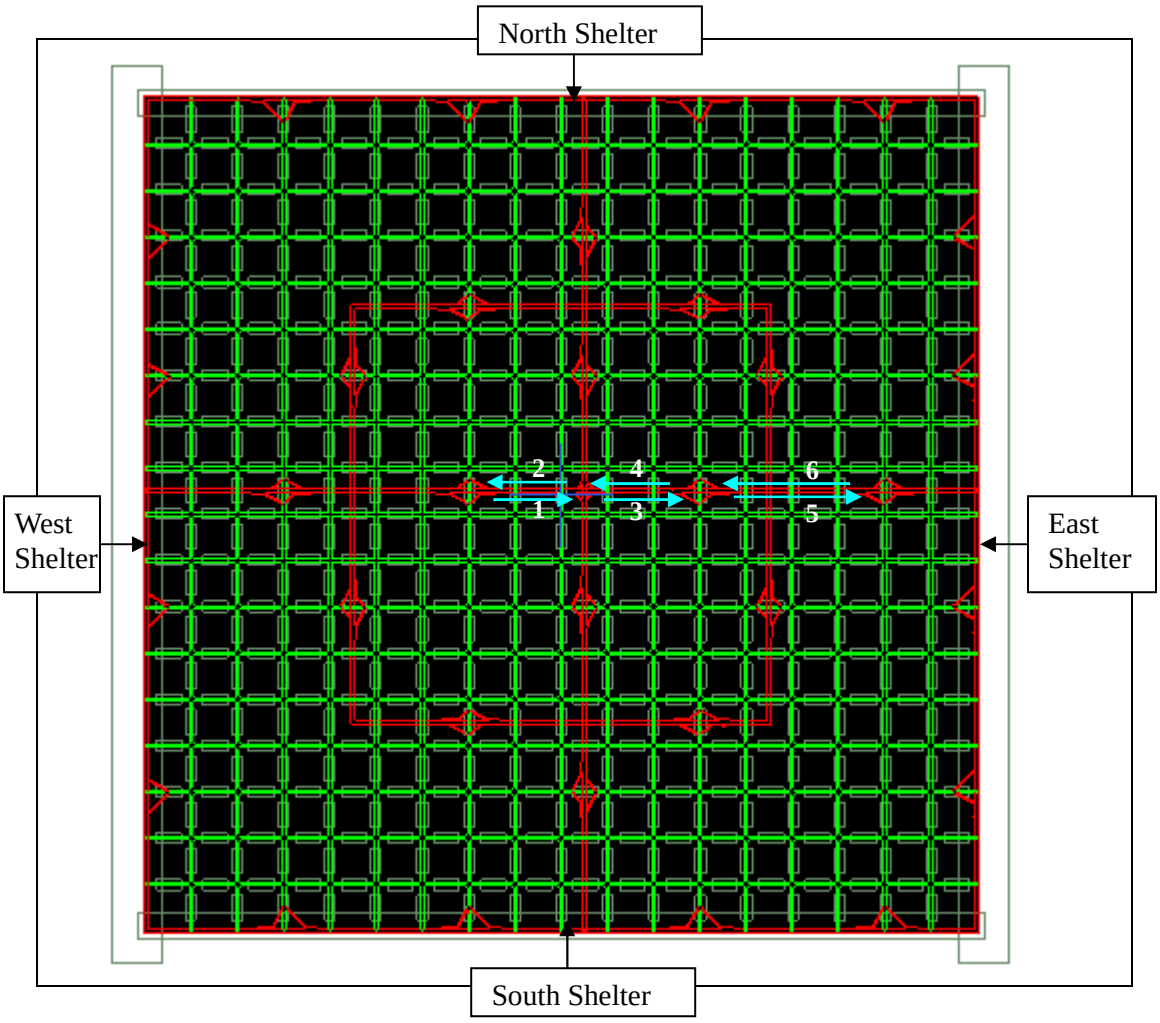


Figure 5-1. Simulated road network with a few picked links

Next, we investigate directional link flows by using the API tool we proposed in chapter 3. The simulation results are generated for the three scenarios, which include i) no family consolidation, ii) 50% of families consolidated, and iii) all families consolidate before the evacuation. We identified a few freeway links that carry significant volumes as shown in blue in Figure 5-1. Links 1, 3, and 5 are eastbound links and links 2, 4, and 6 are westbound ones.

5.2 Study Results on Links #1 and #2

In this part, we study the directional flows on the pair of links located on the west of the network. For the pair of links 1 and 2, the **westbound** direction is the direction to leave the network. Figures 5-2, 5-3, and 5-4 show the traffic flow on links 1 and 2 under the 0%, 50%, and 100% consolidation rates. The blue bars represent traffic flows on the eastbound link, and the pink bars the westbound flows. As shown, for the three consolidation scenarios, the westbound flows are heavier than the eastbound ones. Not surprisingly, the 100% consolidation scenario has the highest eastbound flows, which represent significant numbers of vehicles traveling contrary to the primary evacuation direction in order to consolidate. It should also be noted that the evacuation times increase with the increasing percentage of consolidation rates.

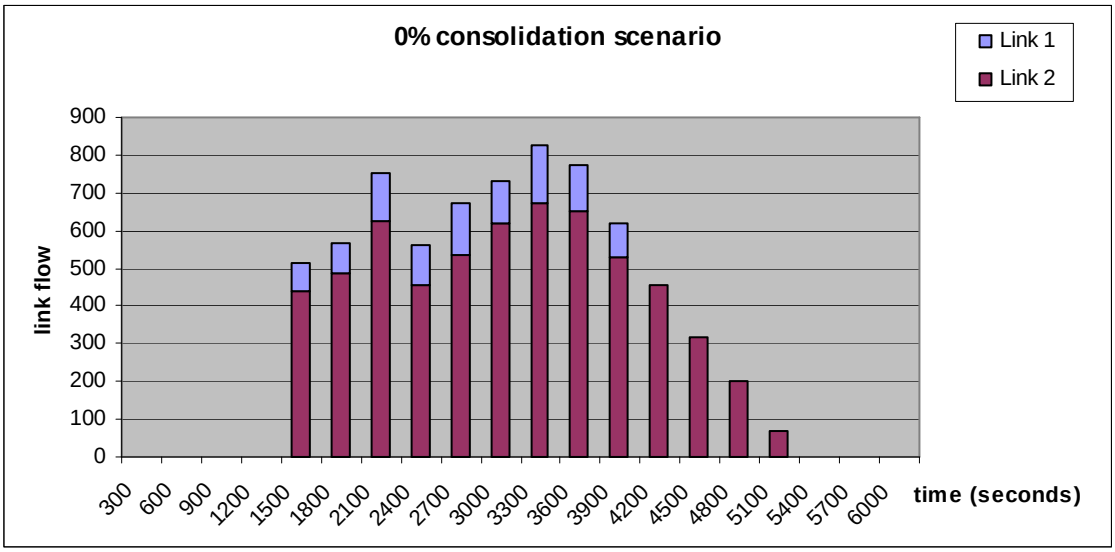


Figure 5-2. Directional flow counts on link 1 and 2 under 0% consolidated scenario

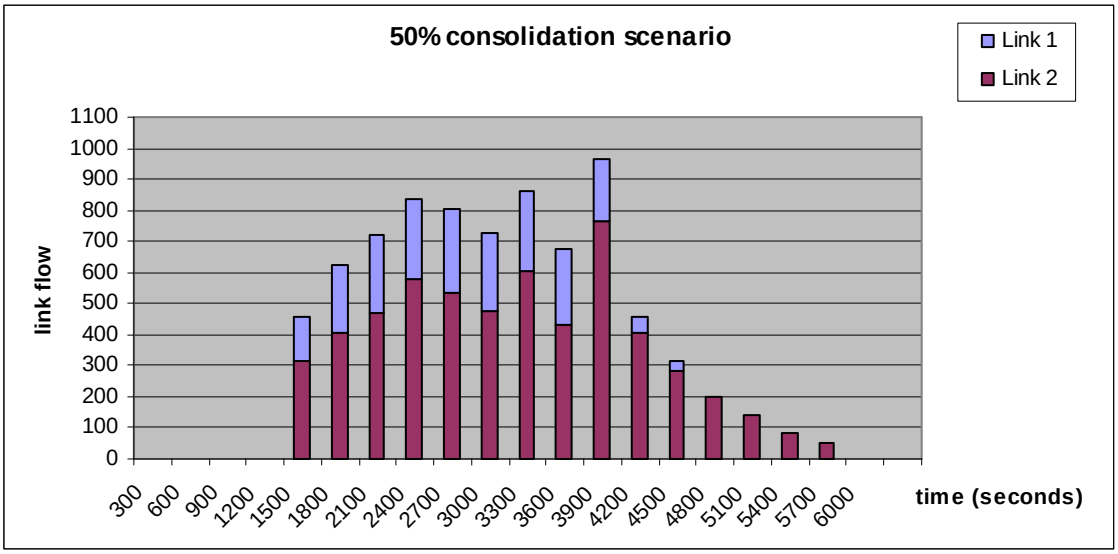


Figure 5-3. Directional flow counts on link 1 and 2 under 50% consolidated scenario

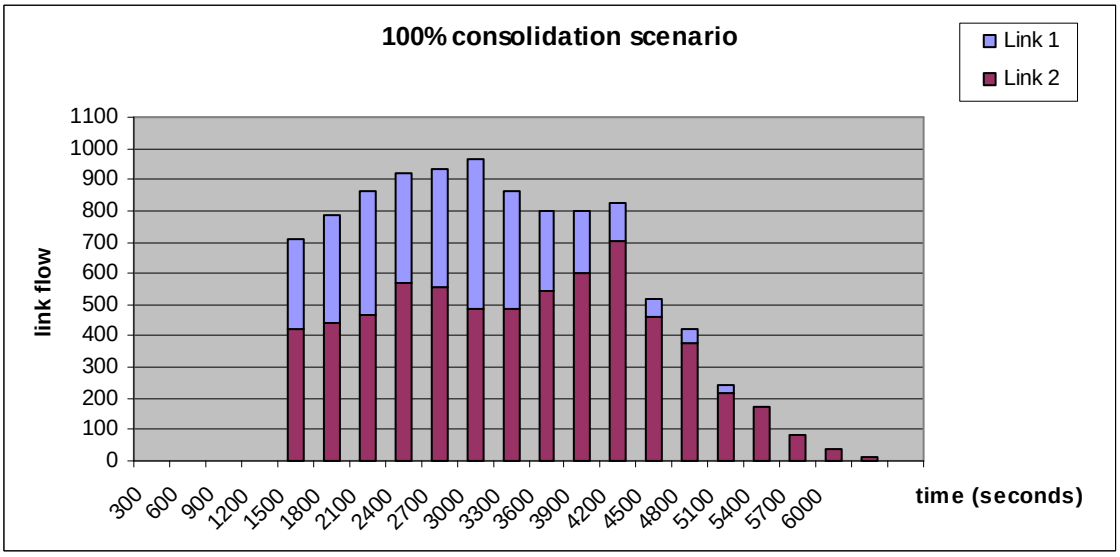


Figure 5-4. Directional flow counts on link 1 and 2 under 100% consolidated scenario

These findings reveal that as the consolidation rates increase, inbound traffic continues to increase for some time after the initiation of the evacuation, which leads to the longer evacuation time.

As mentioned earlier, one of the most popular evacuation strategies is the contraflow operation, which converts one or more of the inbound lanes to outbound ones. In this study, as documented in these figures, the appropriate turning point to reverse eastbound lanes at link 1 to westbound for Scenario 1 is about 4200 seconds, as some vehicles try to turn around using the freeway link when the evacuation starts. For Scenario 2 the appropriate time is about 4800 seconds, and for Scenario 3 it is about 5400 seconds. Thus, this study shows that the consideration of full consolidation could delay the turning point in a contraflow operation by about 30 percent compared with no consolidation. This is an interesting finding as the inappropriate implementation of contraflow strategy may cause deterioration of the network congestion during the evacuation.

5.2 Study Results on Links #3 and #4

In this part, we study the directional traffic flows on the first pair of links located on the east of the network. This pair of links, i.e. link 3 and 4, as shown in Figure 5-1, is located at the east section of the studied network, and thus the eastbound direction is the primary direction to leave the network. The study of links 3 and 4 produces symmetrical results as those from the link 1 and 2, as might be expected.

Figures 5-5, 5-6, and 5-7 show the network performance under three different consolidation scenarios, i.e. 0%, 50% and 100% consolidations. In this case, we have the blue blocks represent traffic flows on the westbound link, and the pink blocks the eastbound flows.

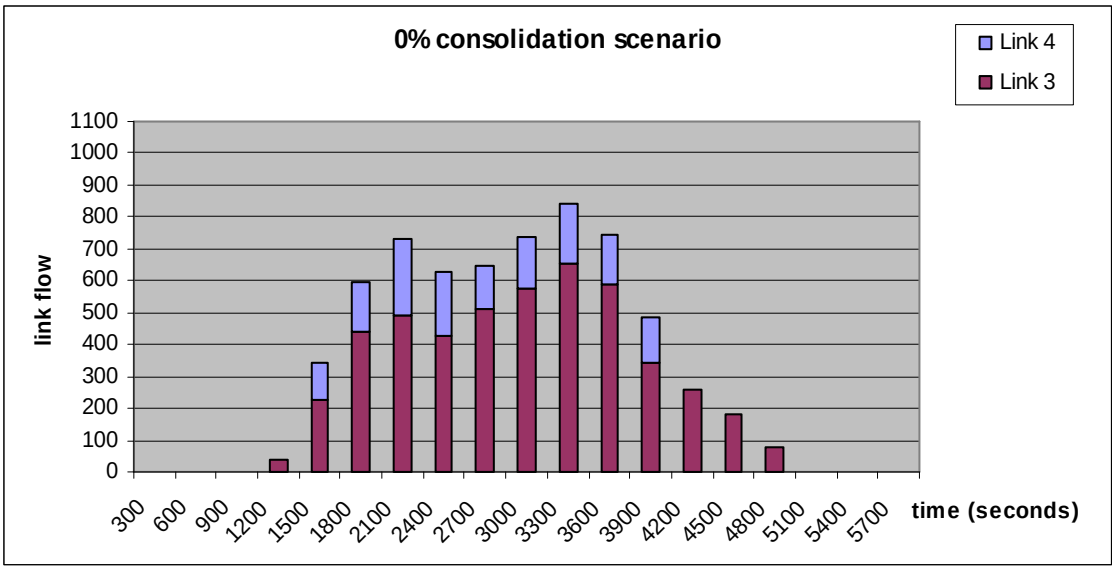


Figure 5-5. Directional flow counts on link 3 and 4 under 0% consolidated scenario

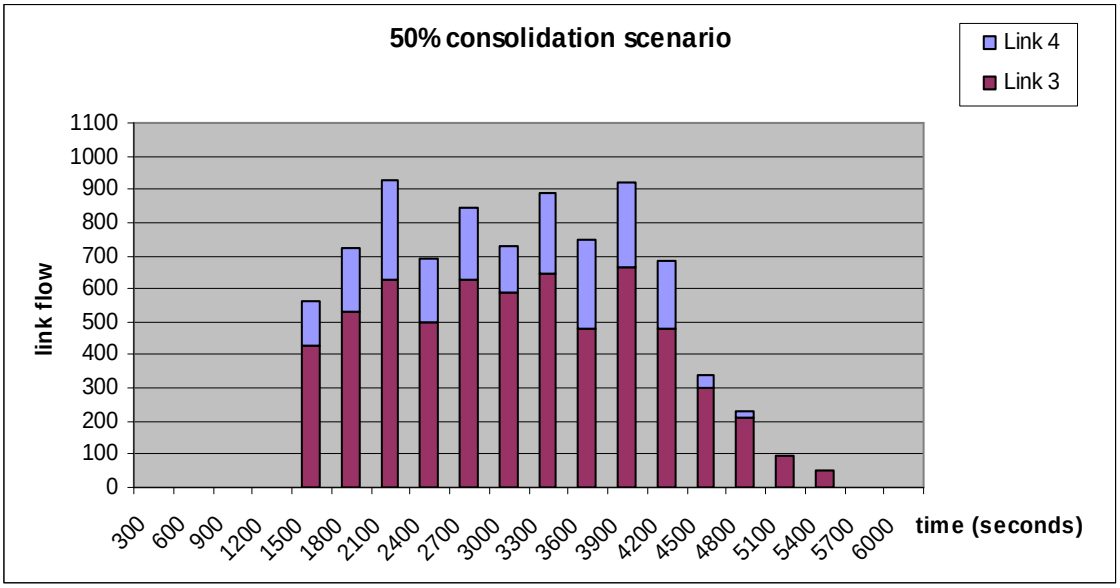


Figure 5-6. Directional flow counts on link 3 and 4 under 50% consolidated scenario

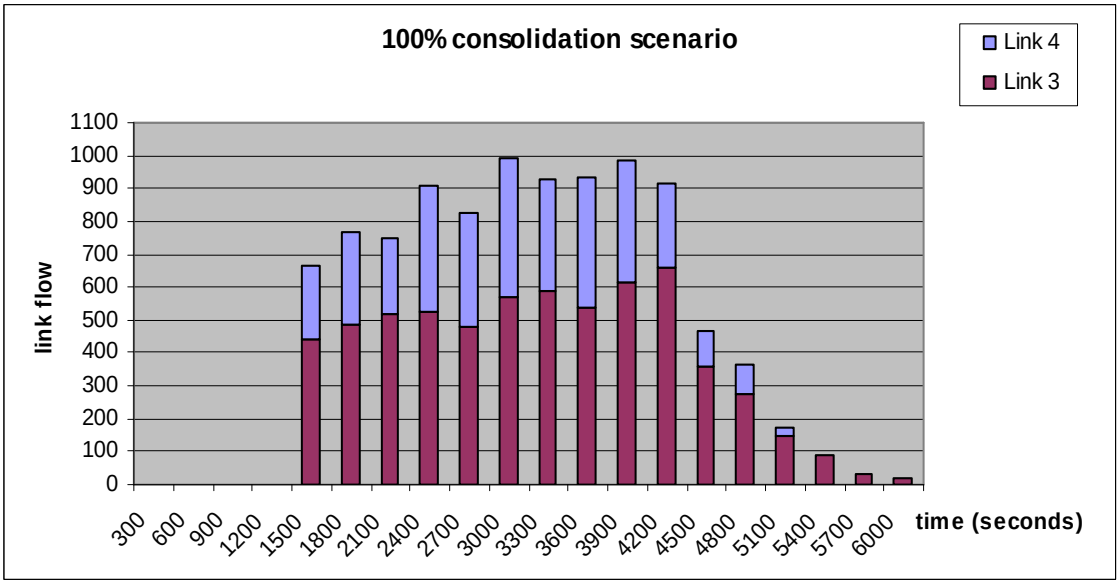


Figure 5-7. Directional flow counts on link 3 and 4 under 100% consolidated scenario

As displayed in these figures, with the increase of the consolidation rates, the westbound flows grow higher and the total evacuate time turns longer. This trend is consistent with what was observed with the western links. In particular, quantitatively, the turning point

to reverse westbound lanes at link 4 to eastbound is 4200, 5100, and 5400 seconds for the Scenario 1, 2 and 3 correspondingly. This study also indicates there is about 30% delay in determining the turning moment of a contraflow strategy when full consolidation is considered. These persistent results are anticipated, because as shown in the Figure 5-1, the pair of links 1 and 2 is symmetrical with the pair of links 3 and 4 in the network. Except for some stochastic variation, the two results are identical to each other.

5.3 Study Results on Links #5 and #6

This section studies the second pair of links located on the east of the network. In this case, the eastbound direction is still the direction to leave the network, as this pair of links 5 and 6, is located at the east quarter of the network. Comparing with the previous pair of links 3 and 4, this pair of links is one step closer to the shelter, as shown in Figure 5-1.

The simulation results of directional flow on this pair of links for the three different consolidation scenarios (i.e. 0%, 50% and 100% consolidation rates respectively) are displayed in Figures 5-8, 5-9, and 5-10 respectively. As shown, the results for the turning point in this case are about 4200, 4500, and 4800 seconds for the Scenario 1, 2 and 3 respectively. That is to say, high consolidation rates could delay the turning point to reverse the inbound lanes to outbound.

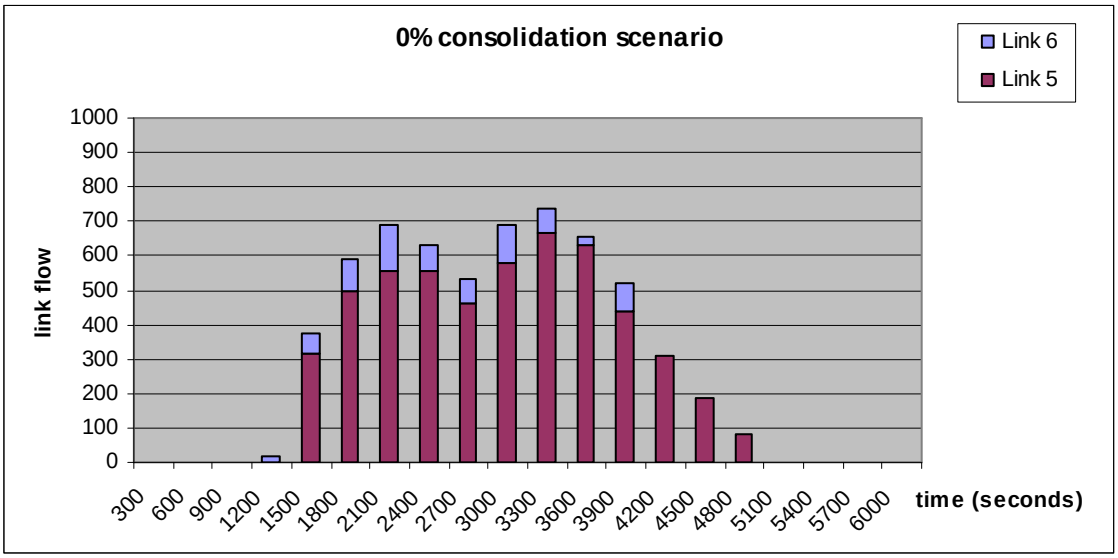


Figure 5-8. Directional flow counts on link 5 and 6 under 0% consolidated scenario

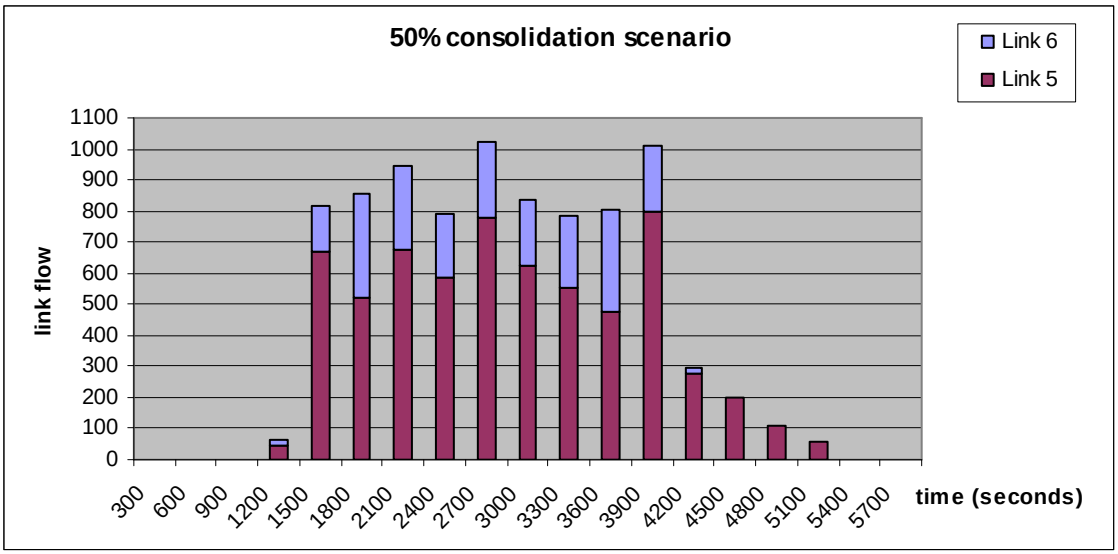


Figure 5-9. Directional flow counts on link 5 and 6 under 50% consolidated scenario

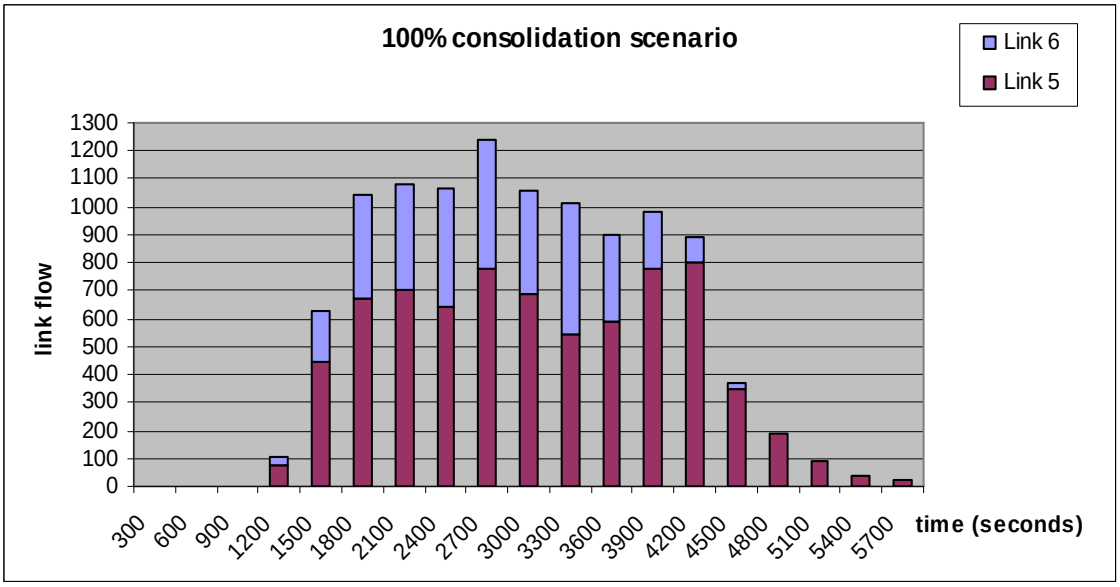


Figure 5-10. Directional flow counts on link 5 and 6 under 100% consolidated scenario

Results from these figures reveal that, comparing with the previous links, this set of links has earlier turning points and heavier outbound link flows. This is because, given the geometry of the test network, the traffic at the outside of the network is heavier than

the inside once the evacuation begins. Besides, the household consolidation behaviors lead to a substantial portion of inbound traffic at the beginning of the evacuation.

To further investigate this issue, a comparison is made between each of the directional links from the two pairs of links that are located at the east quarter of the network. Namely, we compare the eastbound flows between link 3 and link 5, and the westbound flows on link 4 and link 6.

Figures 5-11, 5-12, and 5-13 show the link flows in the eastbound direction for all three consolidation scenarios, i.e. 0%, 50%, and 100% consolidation rate, respectively. As shown, overall, the outer link 5 that is closer to the network boundary (i.e. the shelter) carries more traffic than the inner link 3. As the consolidation rate increases, there are significant differences of link flows between link 3 and link 5. The outer link carries more outbound flow than the inner link at the beginning of the simulation. Later on, with more consolidated vehicles into the network, the inner link flows turn heavier.

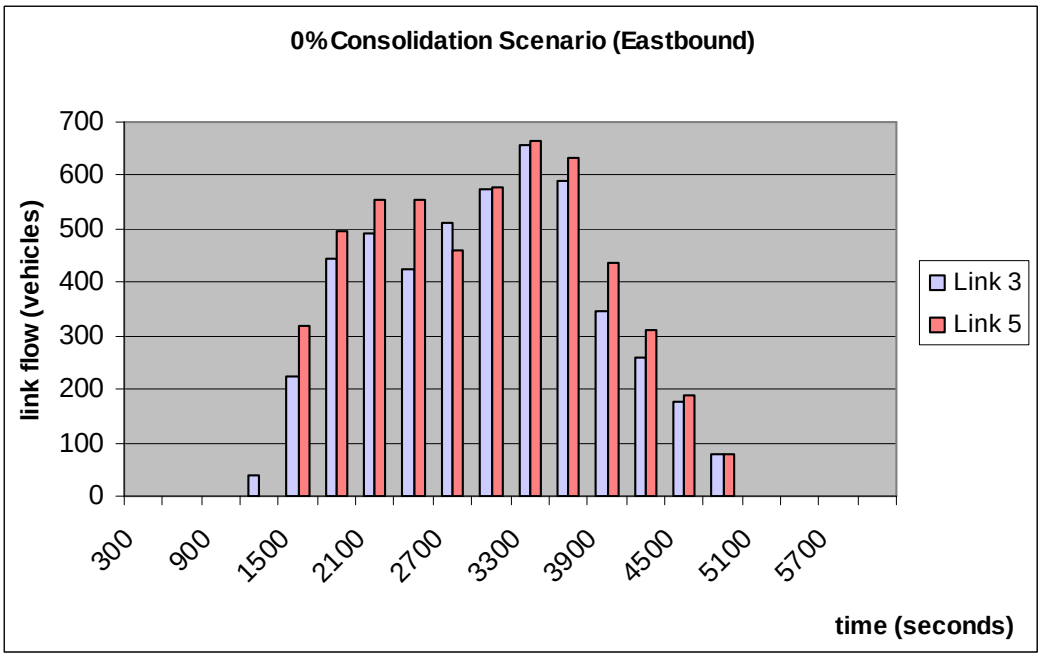


Figure 5-11. Directional flow counts on eastbound links under 0% consolidated scenario

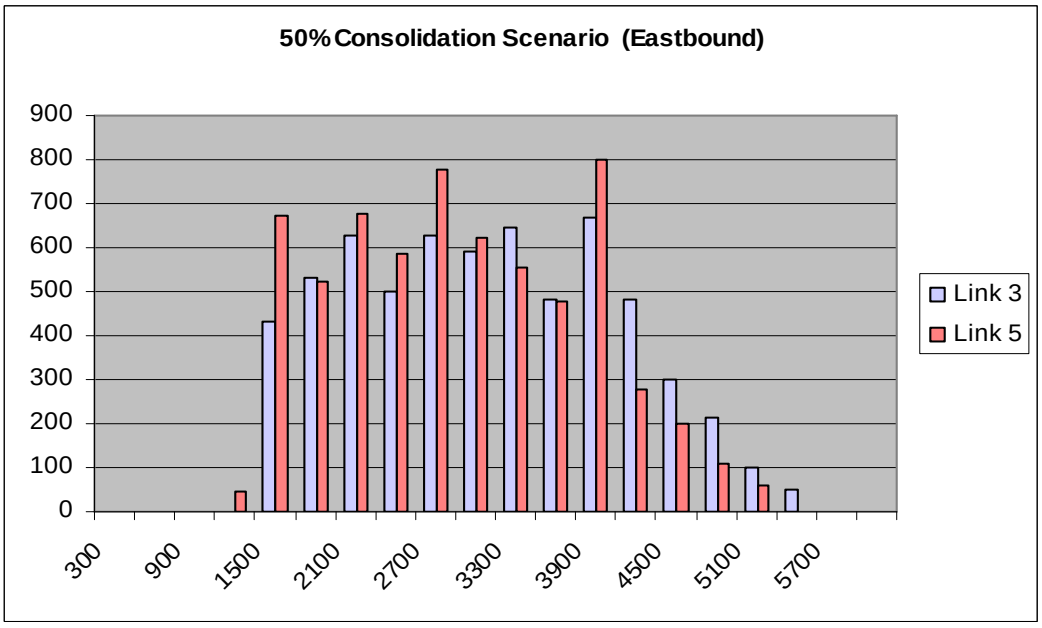


Figure 5-12. Directional flow counts on eastbound links under 50% consolidated scenario

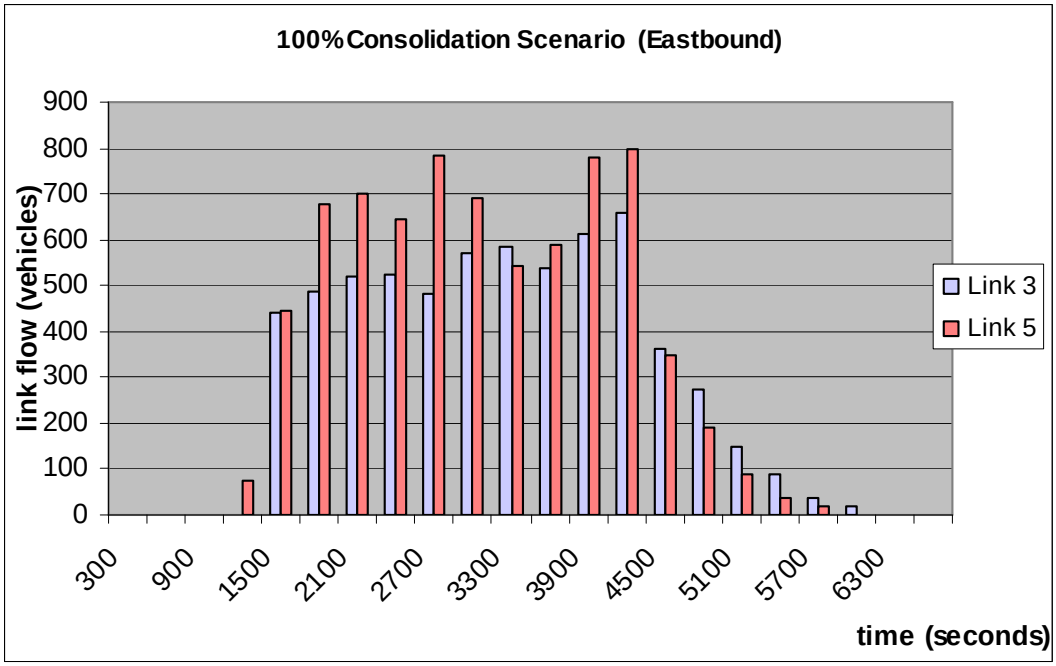


Figure 5-13. Directional flow counts on eastbound links under 100% consolidated scenario

The link flows on westbound links 4 and link 6 for the three different consolidation scenarios are shown in Figures 5-14, 5-15, and 5-16 respectively. As shown, with 0% consolidation, there are fewer vehicles on link 6, which is an inbound link at about 65% of the distance from the population center. As the consolidation by household rate increases, the inbound flows on both links increase significantly. Later on, closer to the end of the evacuation, fewer inbound vehicles are on the outer link 6.

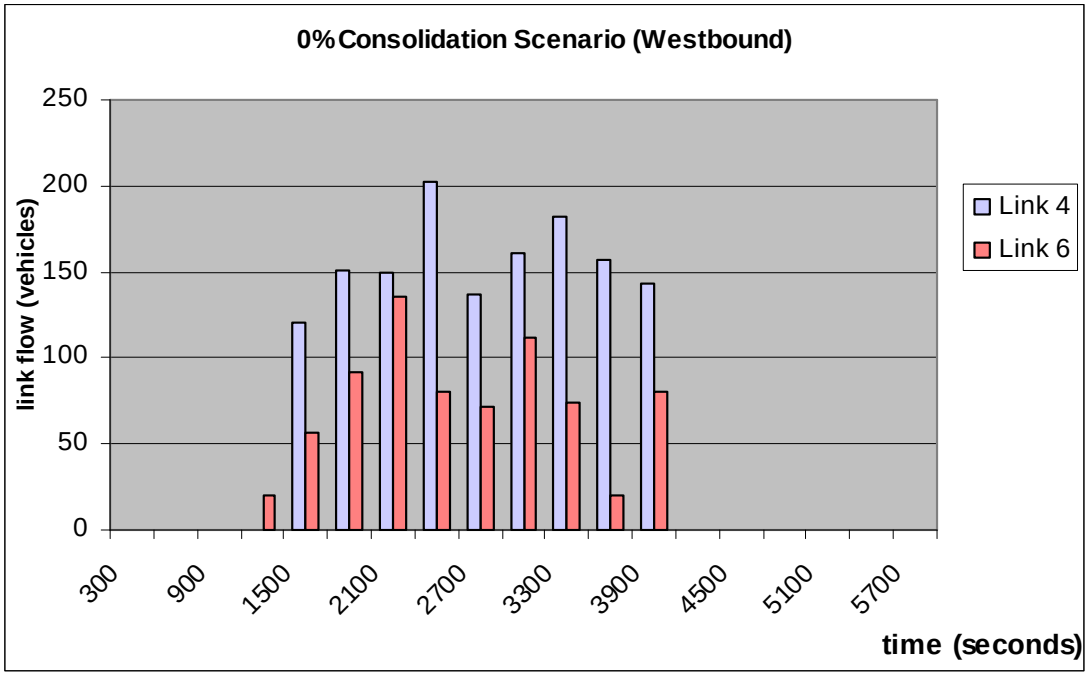


Figure 5-14. Directional flow counts on westbound links under 0% consolidated scenario

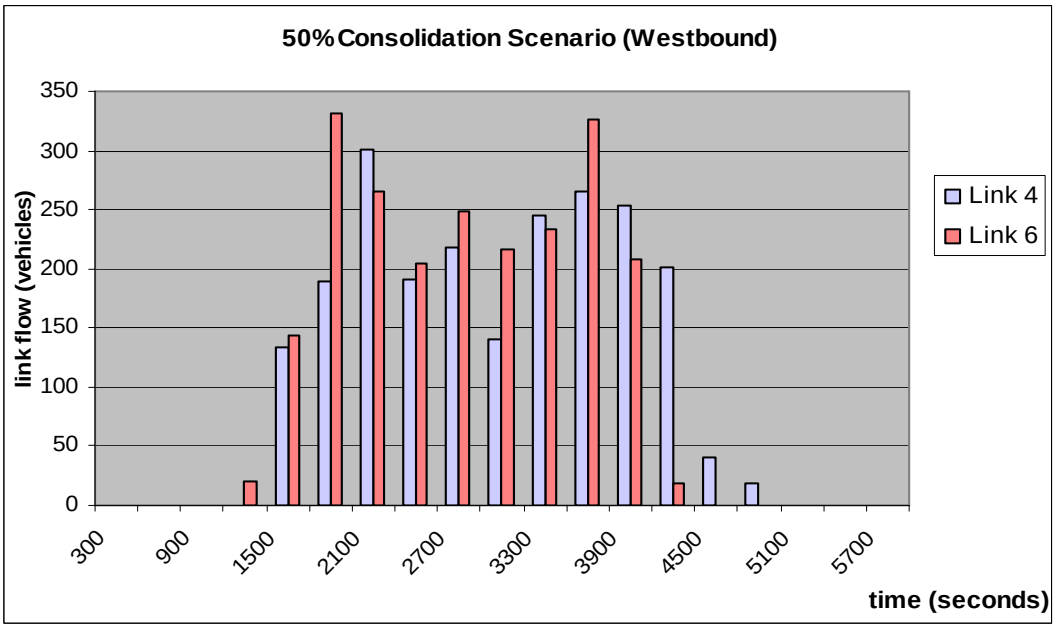


Figure 5-15. Directional flow counts on westbound links under 50% consolidated scenario

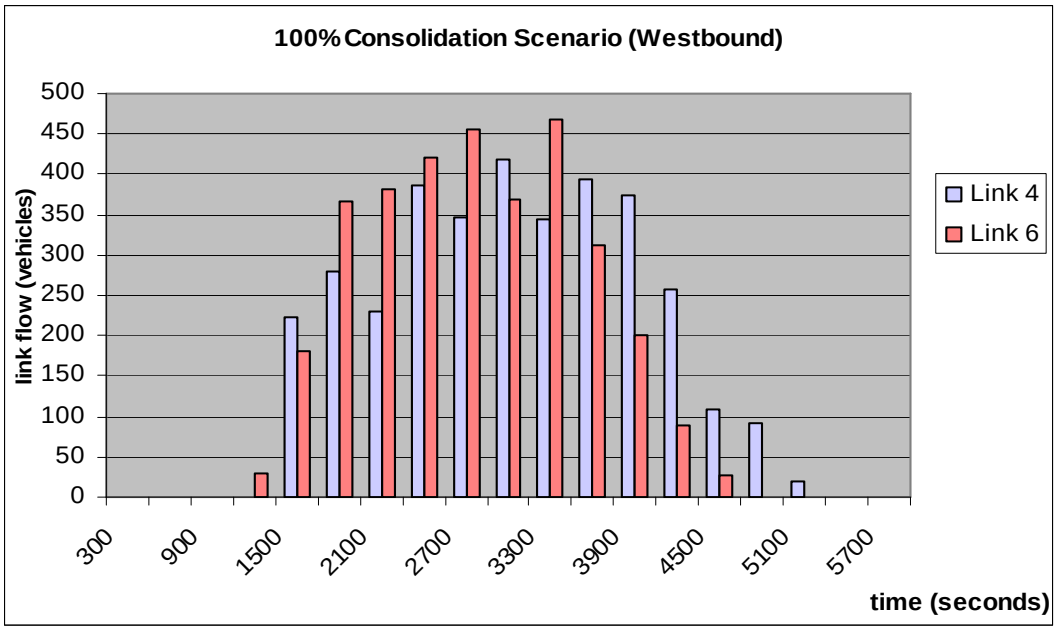


Figure 5-16. Directional flow counts on westbound links under 100% consolidated scenario

Therefore, with the traffic flow pattern disclosed by this study, a proper sequence of reversing lanes, if staged evacuation is considered, is from the outer side of the network to the inner side. In this way, it can help reduce the bottleneck and chaos in the network that would be caused by the consolidation by household behavior in the period immediately following the disaster.

Chapter 6: Information Dissemination and Evacuation Awareness

In this part, we examine the evacuation dynamics with household consolidation under extended conditions, which include: (1) how the efficiency of evacuation information is disseminated, and (2) how much preparedness time the evacuees have for the evacuation.

6.1 Information Dissemination Delays

In this study, we investigate the influence of the delay of evacuation information dissemination, namely, how long it takes for the evacuation order to reach each individual. As noted in chapter 3, there are multiple classes of drivers who are in a number of different states on the studied network when the evacuation starts. There were three sub-models developed, i.e., the sub-model of yet-to-be-released vehicles, the sub-model of en route vehicles, and the sub-model of arrived vehicles (corresponding to the sections 3.6, 3.7, and 3.9). Vehicles in various sub-models handle information delays differently.

As studied in the previous section, the same network model is used with high demand flows, i.e., the network is seeded with 5,000 families and 100,000 vehicles. Evacuation is also assumed to start after a quarter of an hour of the simulation with low traffic volumes. The time for a percentage of vehicles to arrive at the shelters or other destinations, and the cumulative population to be evacuated, are also used as measures of effectiveness for evacuation performance in this study.

We study the evacuation information dissemination delays in this part with 0% consolidation rate. The delays consist of the awareness time and revision time, which are presented in details in section 3.6 and 3.7 respectively. Generally, the awareness time represent the delays for a yet-to-be-released vehicle to receive the information about the evacuation, while the revised time stands for the information delays for those vehicles that have to change their destination en route due to the evacuation. Per the discussion in chapter 3, these delay times vary by each person's location, his/her planned destination, his/her current activity, and other issues such as whether or not the person has to consolidate with other family members. In this case study, we assume that both the awareness time and revised time are random variables with a uniform distribution between 0 and 30 minutes. The mean of the delay time is 15 minutes. This distribution is used only for the illustration purposes. Users can put any specific distribution they want into the API tool.

Simulation results are shown in the Figure 6-1 and 6-2. The blue line represents the network performance without the information delay, and the pink line is with the information delay. It indicates that the "with information delay" curve from the simulation runs has a similar pattern as the "without information delay" curve, but a shift to the right side, compared to the one from the without information delay runs. In other words, with the consideration of information delays, it takes longer to evacuate all evacuees. The reason is that for the without information delay scenario, it simply assumes immediate departures for all the evacuees once the evacuation is ordered. Quantitatively, as shown in Table 6-1, the elapsed time for evacuating 90% of the

population with considering the information delay is 11000 seconds. Comparing with 8400 seconds in the scenario without information delay, it increases about 45 minutes.

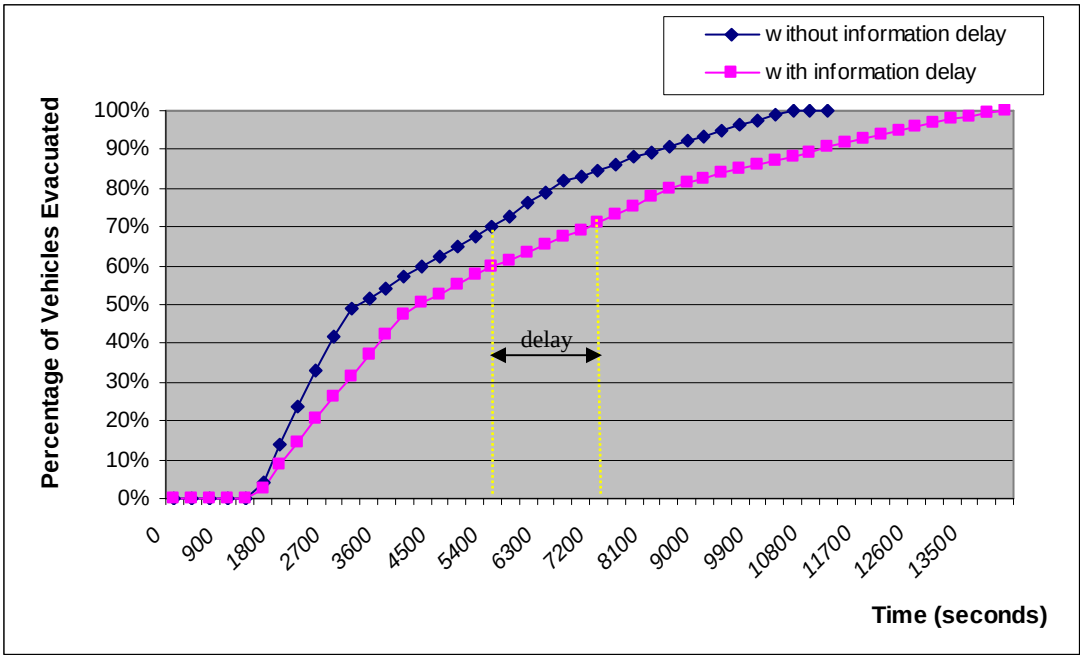


Figure 6-1. Comparison of percentage of vehicles evacuated with/without information delay

Table 6-1. Comparison of times for evacuating different percentage of evacuees with/without information delay

Time (sec)	100%	90%	80%	70%	60%	50%	40%	30%	20%	10%	0%
Without Information Delay	10800	8400	6300	5400	4200	3200	2650	2350	2050	1680	0
With Information Delay	14100	11100	8400	7000	5400	4200	3450	2950	2400	1900	0
Increase	**3300**	**2700**	**2100**	**1600**	**1200**	**1000**	**800**	**600**	**350**	**220**	**0**

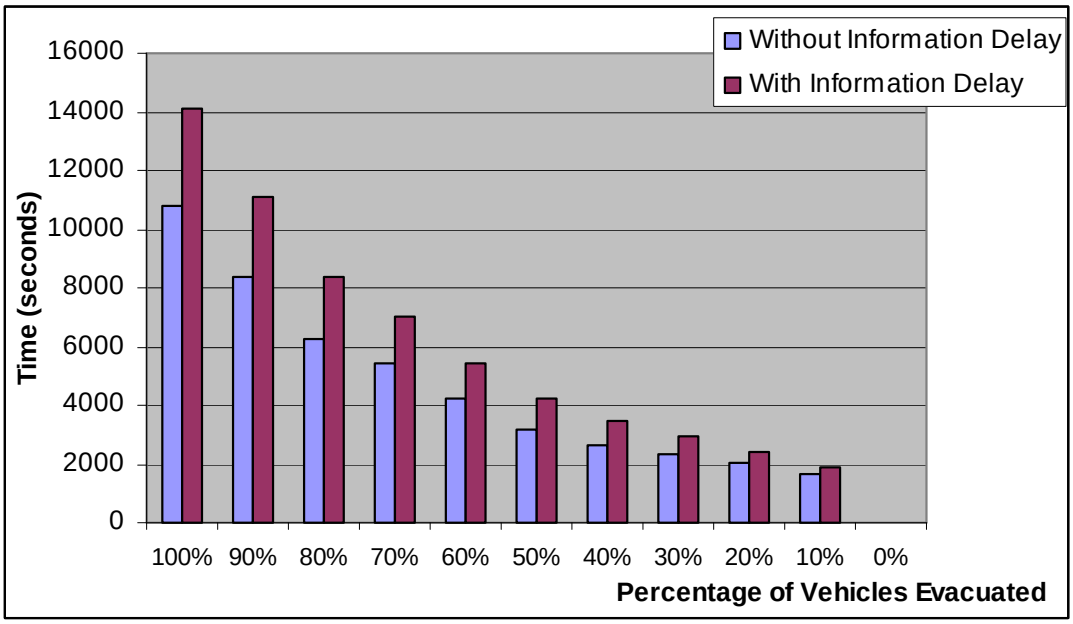


Figure 6-2. Comparison of percentage of vehicles evacuated with/without information delay

From this case study, it can be seen that the evacuation performance metrics that we evaluated were sensitive to the information dissemination delays. Therefore, this is an important parameter to estimate as correctly as possible.

6.2 Evacuation Preparedness

In this section, we study the evacuation preparedness time. We assume that there is a preparedness time for vehicles that arrived at home to prepare for the evacuation. For example, with a family that has more than one independent family member, some members who arrived at home may find out about the evacuation and start packing items while waiting for other members. Once the last family member arrives at home, the whole family can leave together immediately. In other words, the departure time of the new consolidated trip equals the maximum time between the last family member's arrival time and the earliest time at which someone in the family was aware of the evacuation plus an evacuation preparedness time. In this study, the preparedness time is assumed to be a random variable with a uniform distribution between 0 and 30 minutes. The mean of the preparedness time is about 15 minutes. Again, the distribution is only for illustration purposes.

The simulation results are presented in Figure 6-3. It demonstrates the closeness between the results from the "with evacuation preparedness" scenario and those from the "without evacuation preparedness" scenario. In the "without evacuation preparedness" scenario, the departure time of the new consolidated trip simply equals the last family member's arrival time plus a random delay. In the "with evacuation preparedness" scenario, the departure time = max {last family member's arrival time, earliest time at which someone in the family was aware of the evacuation}+delay. The "with preparedness" curve from the with evacuation preparedness scenario is shifted to the left side of the without preparedness scenario. It indicates that, when considering the

preparedness of the evacuation among family members, the evacuees start to evacuate a little bit earlier. The reason is that in the "without evacuation preparedness" scenario, we simply add a random delay time to the family's consolidated trip, however, in the "with evacuation preparedness" scenario, the departure time of the new consolidated trip is the maximum time between the last family member's arrival time and the earliest time at which someone in the family was aware of the evacuation, plus a random preparedness time. In this way, the whole family will not wait until all members' arrival to start preparing for the evacuation, which is more realistic. Thus, it causes that, in the "with evacuation preparedness" scenario, the consolidated families start to evacuate earlier than those in the "without evacuation preparedness" scenario.

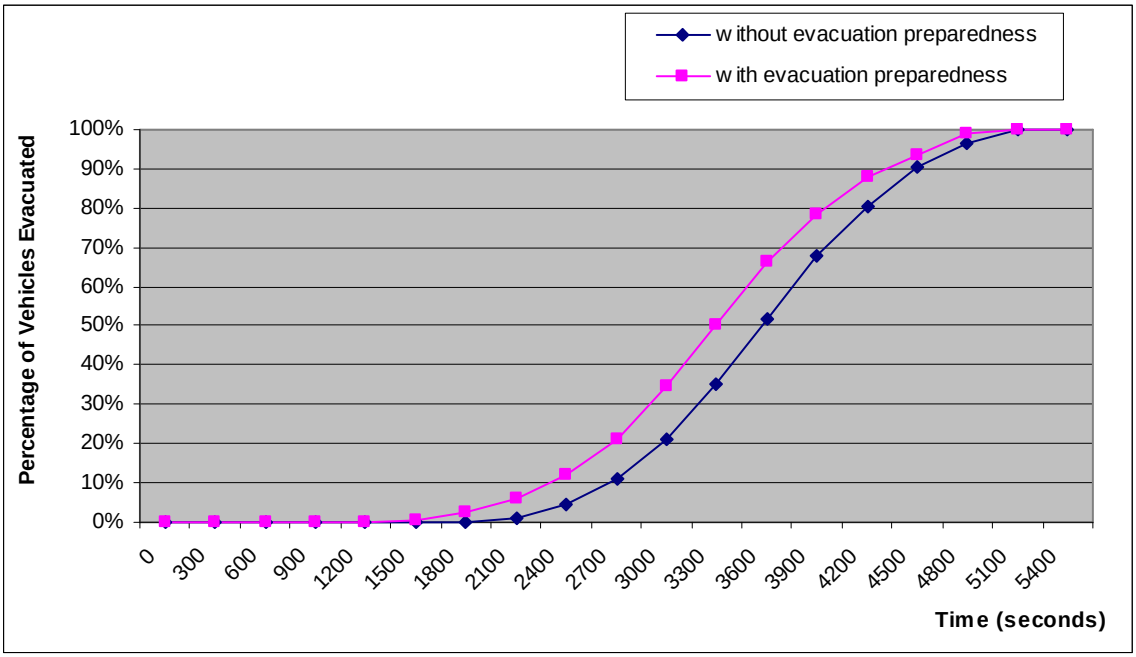


Figure 6-3. Comparison of percentage of vehicles evacuated with/without evacuation preparedness

6.3 Information Dissemination Delays and Evacuation Preparedness

In this part, we consider a scenario that consists of both information dissemination delays and evacuation preparedness simultaneously. In this case, both the information dissemination delay and the preparedness time are random variables between 0 and 30 minutes. The same network model is used with high demand flows, i.e., the network is seeded with 5,000 families and 100,000 vehicles. The purpose of this study is to show that, if there are data for both information delay and the evacuation preparedness time, how the factors affect the evacuation together.

Figure 6-4 gives an illustration of the simulation results between the "without information dissemination delays or evacuation preparedness" scenario and the "with information dissemination delays and evacuation preparedness" scenario. At the beginning of the evacuation, in the network, there are more individual vehicles that respond to the evacuation by taking the shortest path to safety. Therefore, due to the consideration of the possible delays in evacuation information dissemination, it takes longer to evacuate the 10% to 15% of evacuees for the "with information dissemination delays or evacuation preparedness" scenario. Later in the evacuation, when more families who finished their consolidation process enter the network to evacuate, the "with information dissemination delays and evacuation preparedness" curve gradually becomes higher than the other one, as the preparedness time that each family has for the evacuation has been taken into consideration.

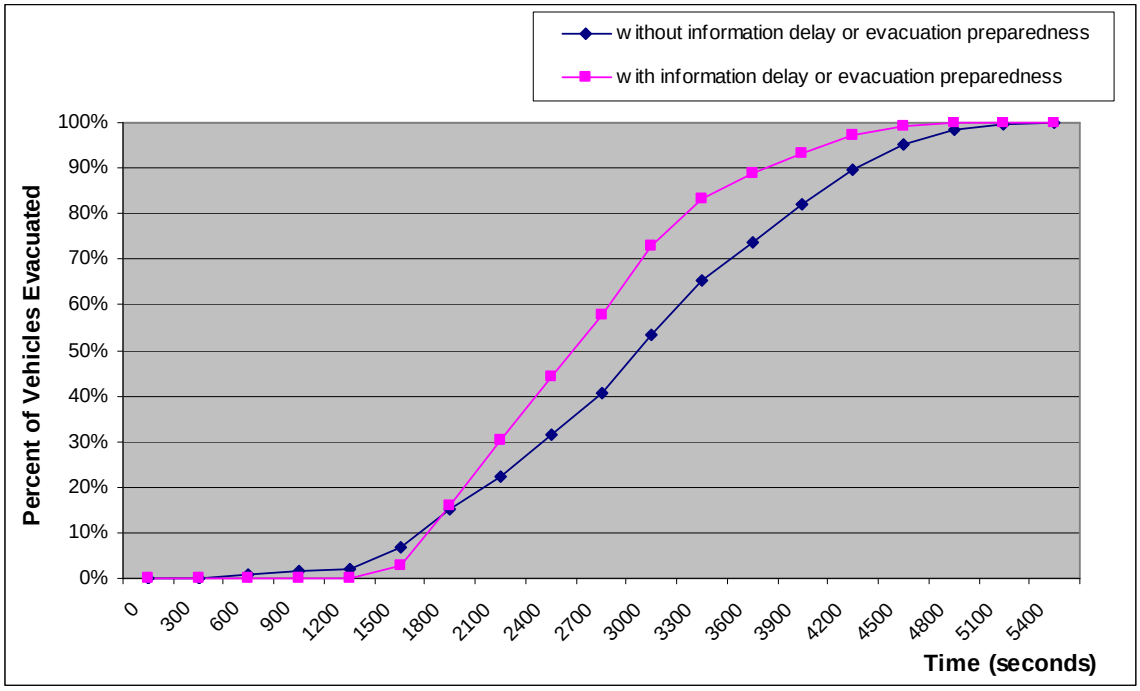


Figure 6-4. Comparison of percentage of vehicles evacuated with/without information dissemination delays + evacuation preparedness

Chapter 7: Demographics and Geography

In the previous chapters, we discuss that some traffic characteristics, such as the traffic demand, information dissemination delay, and the time that evacuees prepare for the evacuation, has an impact on the household consolidation evacuation. There are also some other factors that might affect household consolidation behaviors during an evacuation, such as the number of vehicles in a family, the number and location of shelters in the network, and so on. In this chapter, we perform two studies to investigate demographics and geography issues that affect the evacuation with consolidation.

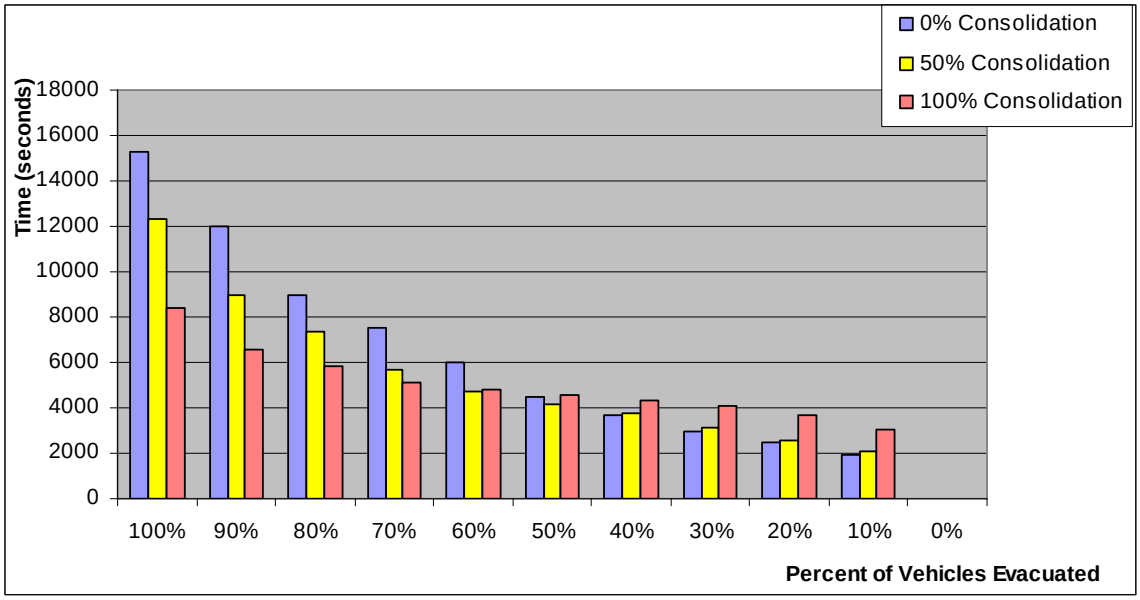
7.1 Distribution of Vehicle Ownership among Households

Different number of vehicles in a family may have different affect on the network performance during an evacuation. In the study we presented in the chapter 4, we adopt the following distribution of vehicle ownership: 50% of the families own a single vehicle, 40% of them own two, and 10% own three. No household owns more than three vehicles. With this distribution, the average households have a number of 1.6 vehicles. On the other hand, according to data from the Bureau of Transportation Statistics (RITA, 2001), the average households have 1.8 drivers. It also indicates that "households with more members are likely to have more personal vehicles available for regular use. For example, single-person household average about one vehicle while households with two members average about two vehicles. However, households with seven or more members average about 2.8 personal vehicles." (RITA, 2001)

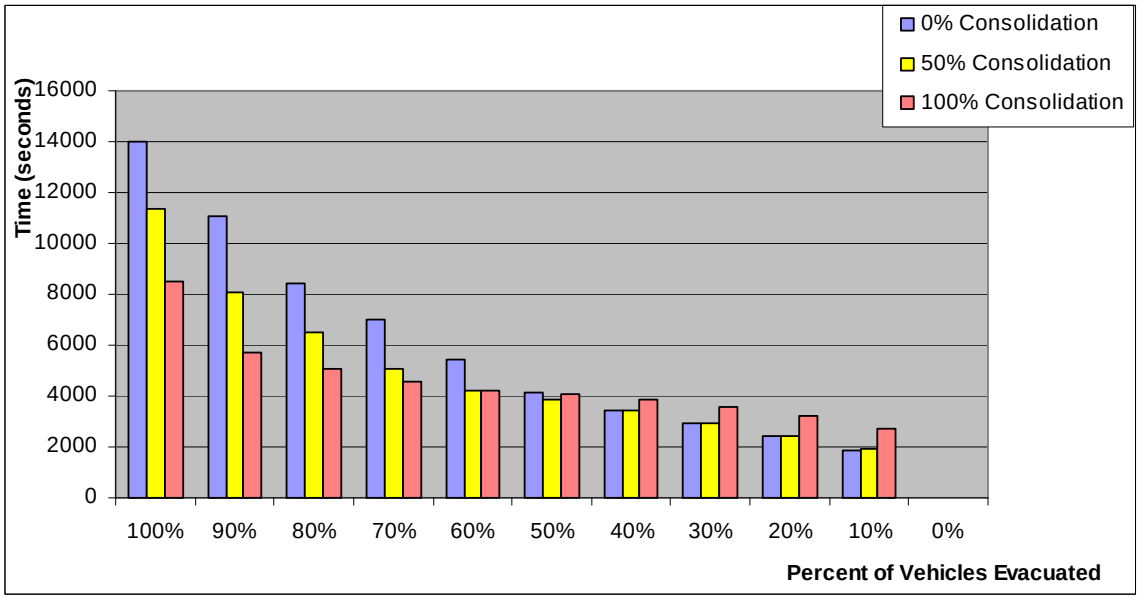
Therefore, in this study, we adjust the distribution of vehicle ownership as: 30% of families have one single vehicle, 50% of them have two vehicles, and 20% have three. The mean number of the household vehicles is 2.2.

We use the same network as the one we described in section 4.1 to test the change of the distribution of the number of vehicles in households. High demand traffics, i.e. 5000 families and 100,000 vehicles, are loaded into the network. The initialization time period is *t=20 minutes* in the simulation, which allows vehicles to fill into the network before the start of the evacuation. Three scenarios about the ratios of family consolidations, namely, 0% consolidation, 50% consolidation, and 100% consolidation, are examined.

Figure 7-1 shows the comparison results between the different distributions of household vehicle ownerships. It is very clear from the figure that the more families have more than two vehicles, the longer evacuation time it takes. For example, under 0% consolidation scenario, for evacuating 90% of population in the case (a) in Figure 7-1, it takes about 12,000 seconds, which is a slight higher than the 11,100 seconds in the case (b) for evacuating the same percentage of evacuees. These numerical results are summarized into table 7-1. This suggests that if the studied area's average household own more vehicles, the expected network clearance time may get longer.



Case (a) - 30% of families own one vehicle, 50% of them own two, and 20% own three



Case (b) – 50% of families own one vehicle, 40% of them own two, and 10% own three

Figure 7-1. Performance of various scenarios on the studied network for vehicle ownership study

Table 7-1. Network clearance time of various scenarios with different distributions of household vehicle ownerships

Case (a) - 30% of families own one vehicle, 50% of them own two, and 20% own three

Time (sec)	100%	90%	80%	70%	60%	50%	40%	30%	20%	10%	0%
0% Consolidation	15300	12000	9000	7500	6000	4500	3700	3000	2450	1950	0
50% Consolidation	12300	9000	7400	5700	4700	4150	3750	3150	2600	2050	0
100% Consolidation	8400	6600	5850	5150	4800	4550	4300	4100	3700	3050	0

Case (b) – 50% of families own one vehicle, 40% of them own two, and 10% own three

Time (sec)	100%	90%	80%	70%	60%	50%	40%	30%	20%	10%	0%
0% Consolidation	14000	11100	8400	7000	5450	4150	3450	2900	2400	1850	0
50% Consolidation	11350	8100	6500	5100	4250	3850	3400	2900	2400	1900	0
100% Consolidation	8500	5700	5100	4600	4250	4050	3850	3600	3200	2700	0

Figure 7-2, 7-3, and 7-4 illustrate how the evacuation times for each scenario of household consolidation vary with different household vehicle ownership distributions. As mentioned previously, we study two sets of distributions:

Case (a) - 30% of families own one vehicle, 50% of them own two, and 20% own three.

Case (b) – 50% of families own one vehicle, 40% of them own two, and 10% own three.

These tends confirm that for all three consolidation scenarios, the households own more vehicles take the longer to evacuate. Furthermore, with all vehicles take the shortest route to the safety immediately, the blue and pink curves almost end at the same time when evacuating 0 to 30 percent of evacuees. However, there are differences in

evacuating low percentage of evacuees in the 50% consolidation and 100% consolidation scenarios. In addition, the more consolidation ratios, the higher differences in the evacuation times.

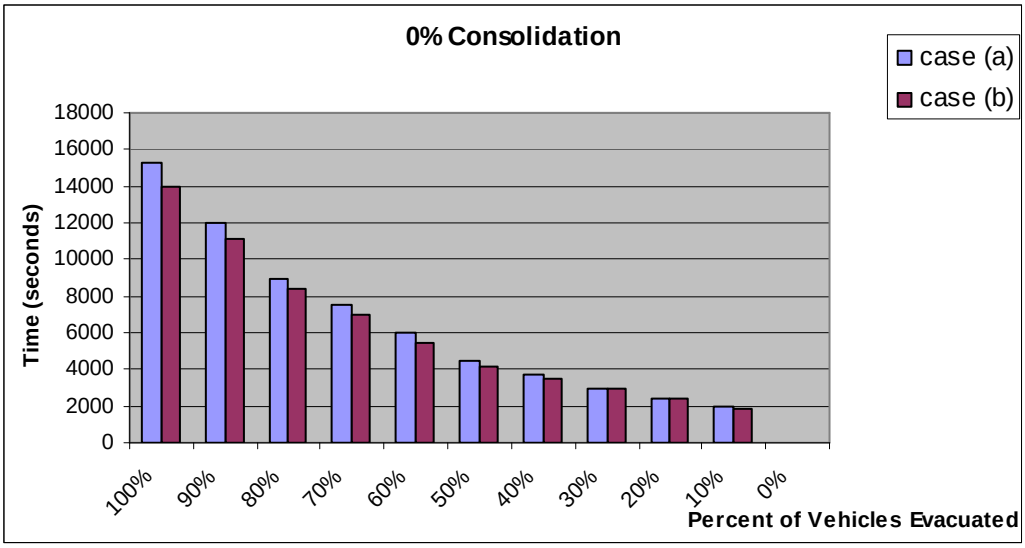


Figure 7-2. Comparison of percent of vehicles evacuated with different household vehicle ownerships under 0% consolidation scenario

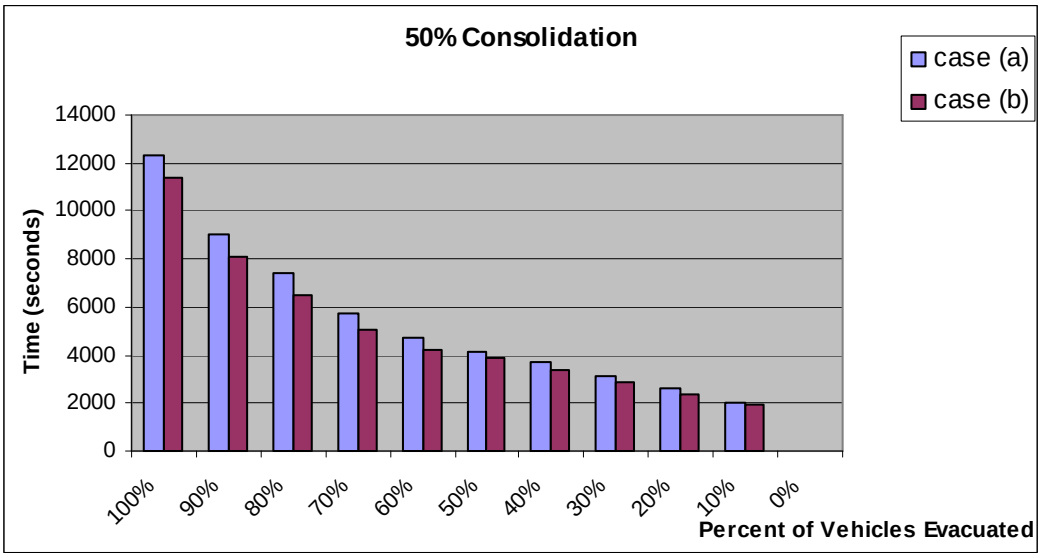


Figure 7-3. Comparison of percent of vehicles evacuated with different household vehicle ownerships under 50% consolidation scenario

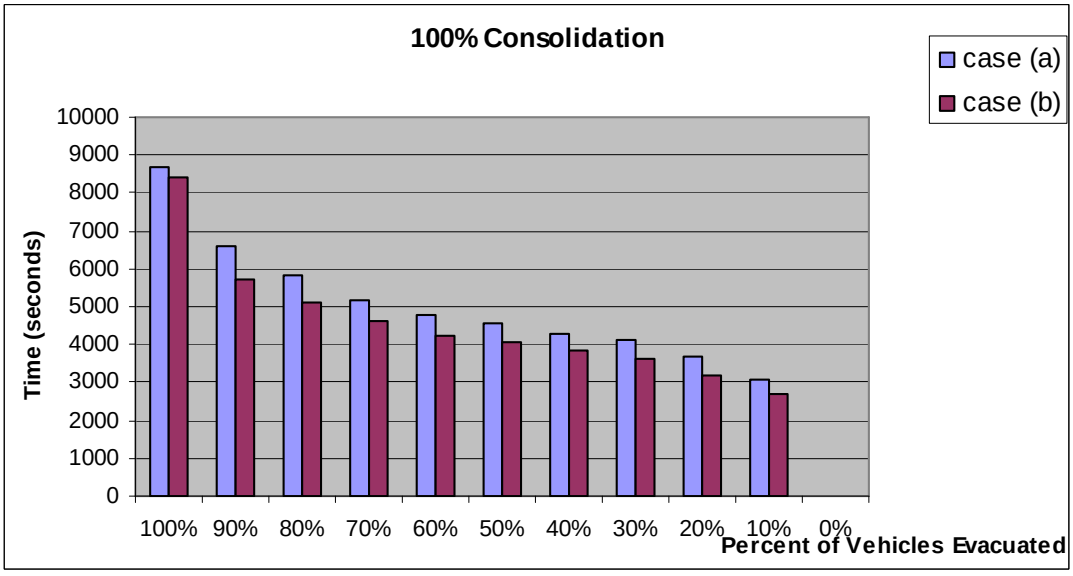


Figure 7-4. Comparison of percent of vehicles evacuated with different household vehicle ownerships under 100% consolidation scenario

7.2 Number of Shelters

The effect of different number of shelters on the network was investigated. In the study we performed in the chapter 4, we assume that there are two shelters that locate in east and south. In this study, we add two more shelters at north and west, as shown in figure 7-5. During the evacuation, vehicles will travel to the closest shelter following certain rules as we described in the chapter 3.

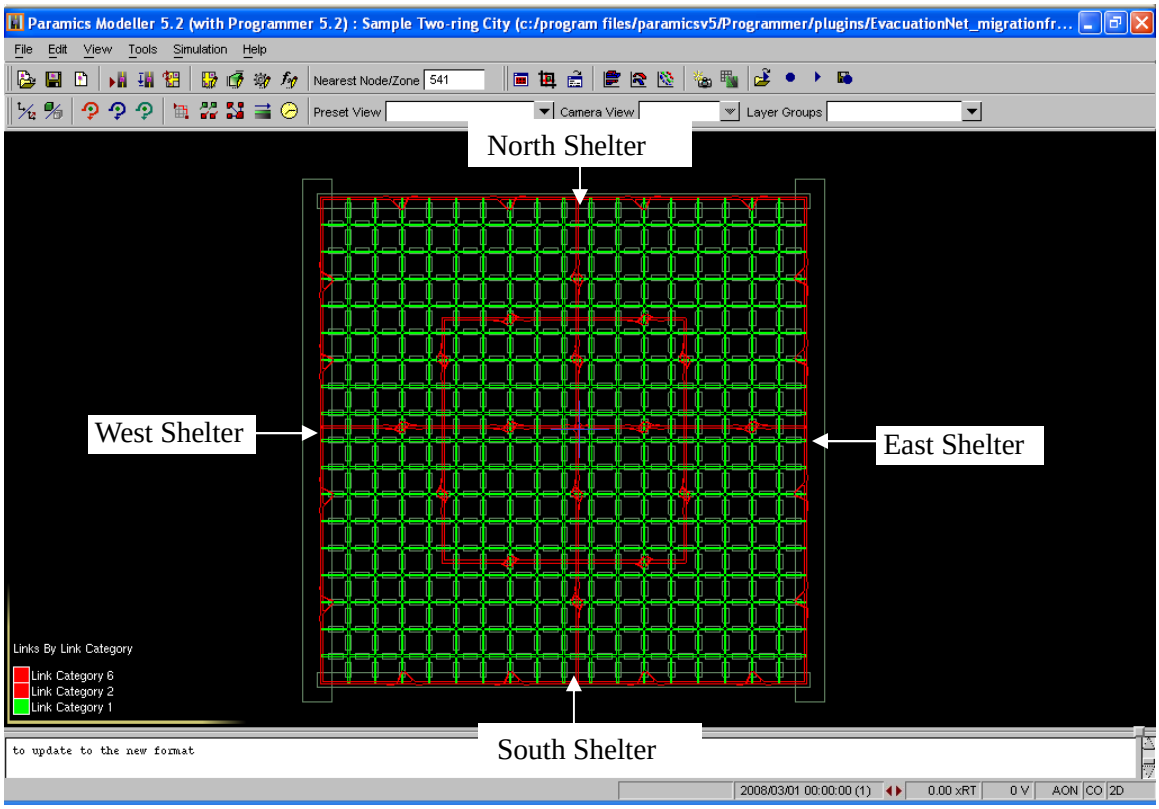
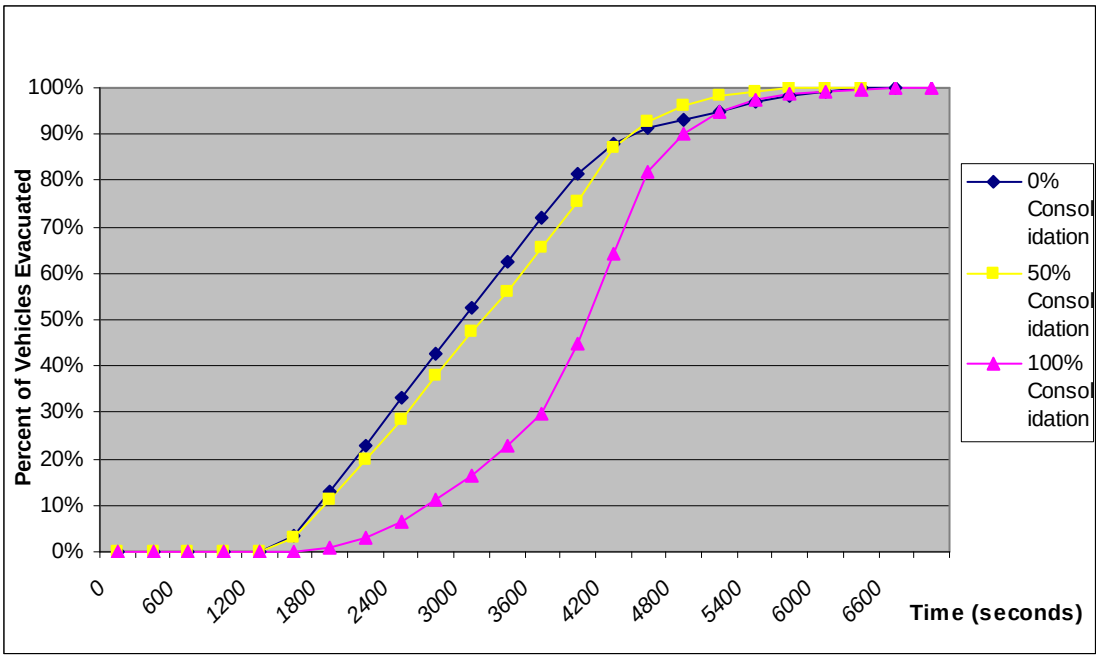


Figure 7-5. Comparison of percent of vehicles evacuated with different household vehicle ownerships under 100% consolidation scenario

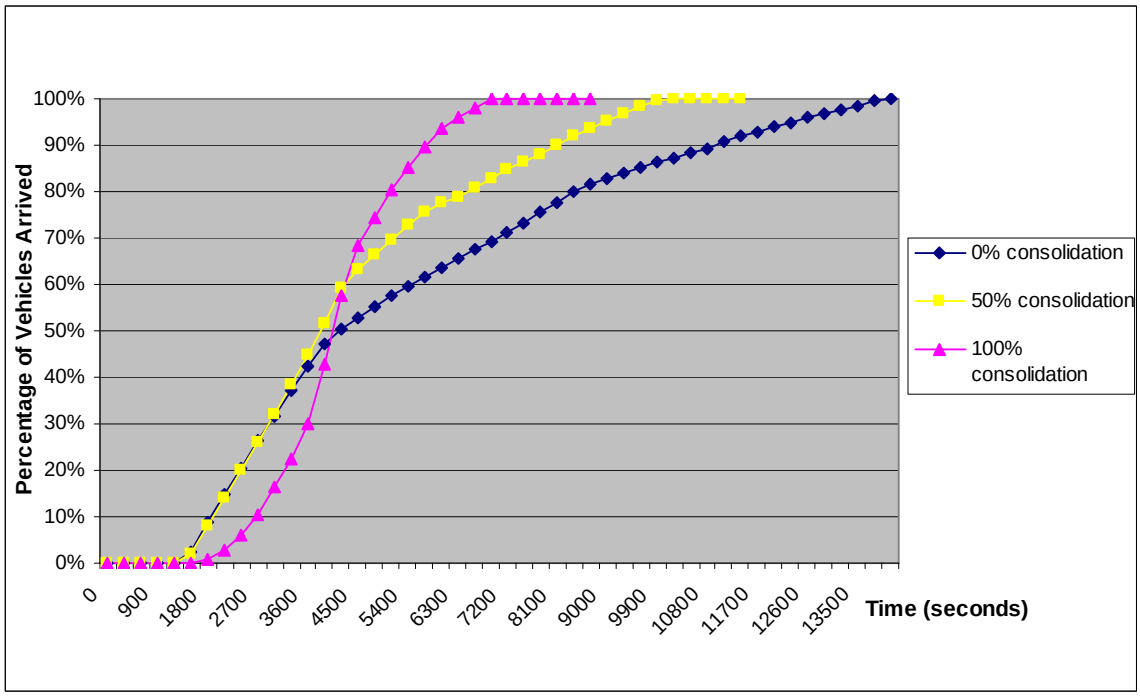
Except for the two more shelters, other network conditions, like high demand, initialization time, etc., are the same as we described in the section 6.1. Different

consolidation by household behaviors, like 0%, 50%, and 100% consolidation rates, are investigated.

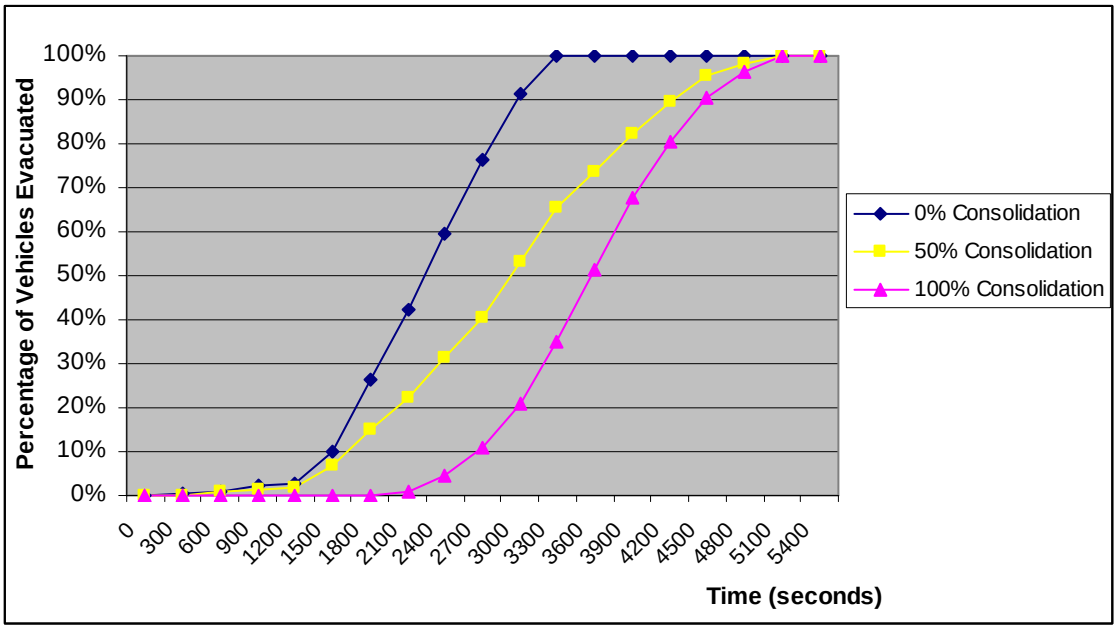
Figure 7-6 and 7-7 shows the comparison results between the four-shelter network and the original two-shelter network. This is an interesting result, because it shows that as the number of shelters increases, the total network congestion decreases dramatically. It indicates that, with four shelters, the network performance with high demand is more likely close to two shelters in low demand situation. The numerical results from the simulations are summarized into Table 7-2.



Case (a) – four shelters with high demand



Case (b) – two shelters with high demand



Case (c) – two shelters with low demand

Figure 7-6. Performance of various scenarios on the studied network for number of shelters study

Table 7-2. Network clearance time of various scenarios with different distributions of household vehicle ownerships

(a) four shelters with high demand

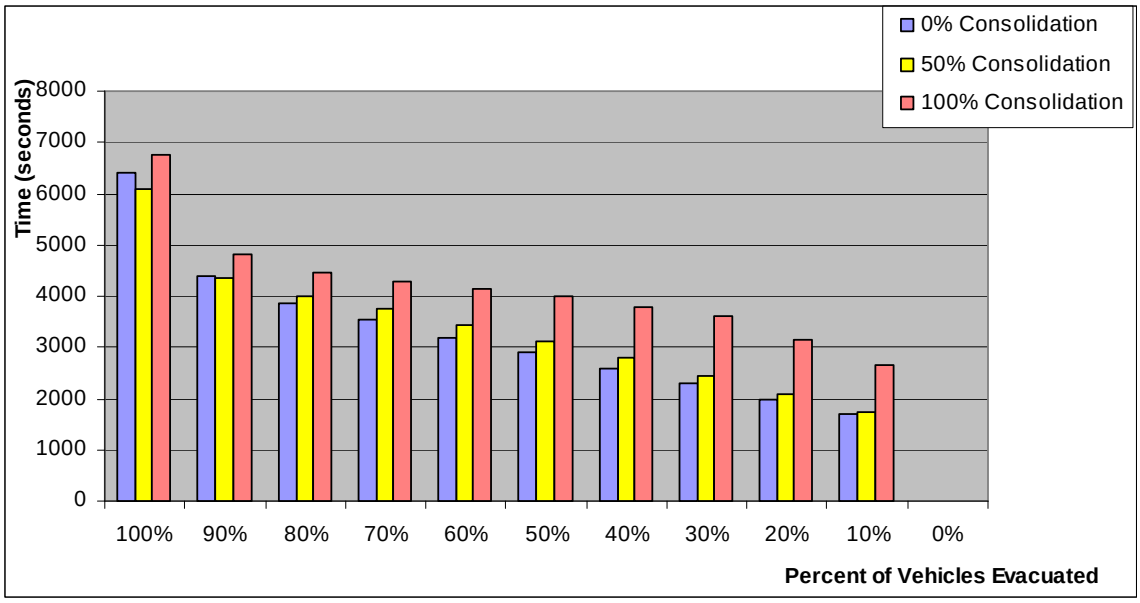
Time (sec)	100%	90%	80%	70%	60%	50%	40%	30%	20%	10%	0%
0% Consolidation	6400	4400	3850	3550	3200	2900	2600	2300	2000	1700	0
50% Consolidation	6100	4350	4000	3750	3450	3100	2800	2450	2100	1750	0
100% Consolidation	6750	4800	4450	4300	4150	4000	3800	3600	3150	2650	0

(b) two shelters with high demand

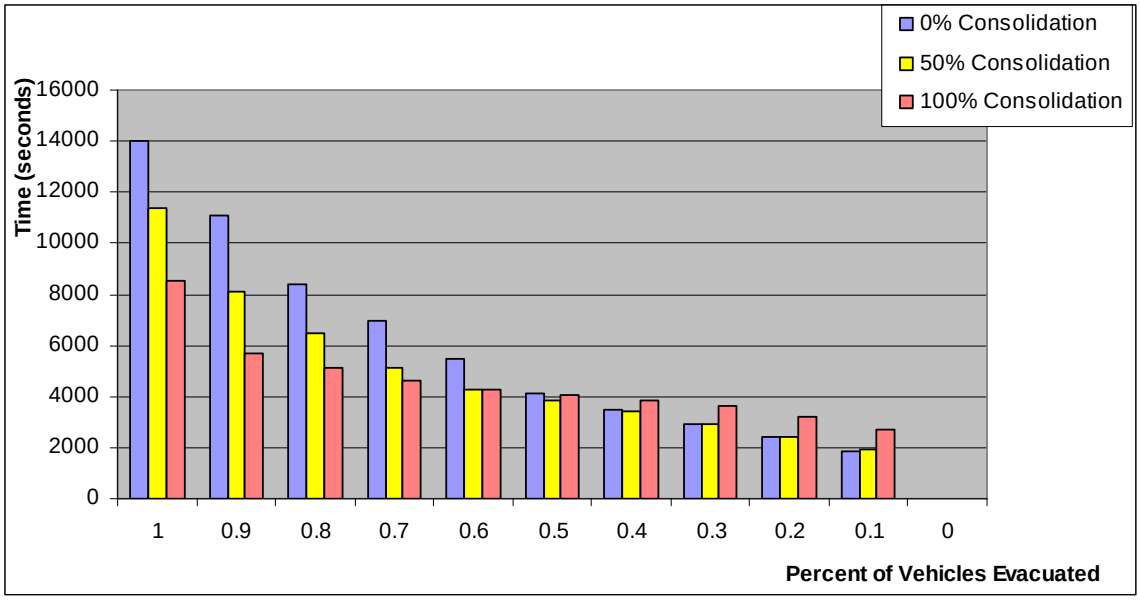
Time (sec)	100%	90%	80%	70%	60%	50%	40%	30%	20%	10%	0%
0% Consolidation	14000	11100	8400	7000	5450	4150	3450	2900	2400	1850	0
50% Consolidation	11350	8100	6500	5100	4250	3850	3400	2900	2400	1900	0
100% Consolidation	8400	5700	5100	4600	4250	4050	3850	3600	3200	2700	0

(c) two shelters with low demand

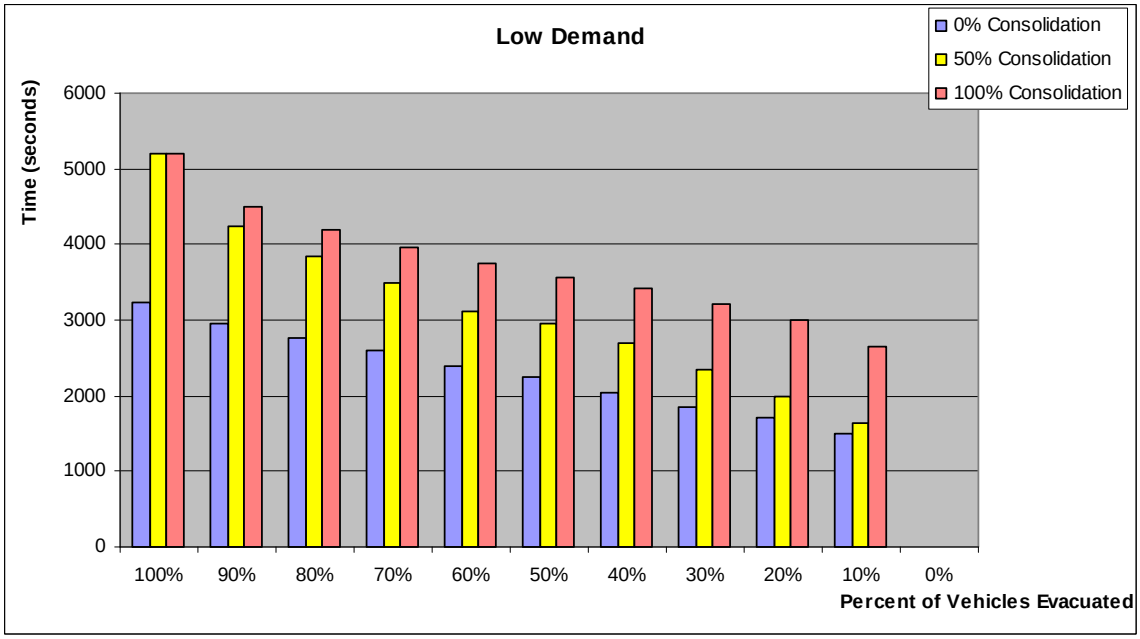
	Scenario 1	Scenario 2	Scenario 3
100%	3240	5200	5200
90%	2960	4500	4250
80%	2760	4200	3850
70%	2600	3950	3500
60%	2400	3750	3120
50%	2250	3560	2950
40%	2040	3420	2700
30%	1860	3200	2350
20%	1700	3000	2000
10%	1500	2660	1650
0%	0	0	0



Case (a) – four shelters with high demand



Case (b) – two shelters with high demand



Case (c) – two shelters with low demand

Figure 7-7. Network clearance time of various scenarios with different number of shelters

Figure 7-8, 7-9 and 7-10 shows the time for evacuating different percentage of vehicles for the three consolidation scenarios, i.e. 0%, 50%, and 100% respectively. The comparison is made among four shelters with high demand, two shelters with high demand, and two shelters with low demand. The results show that, for all consolidation scenarios, there are significant differences between four shelters and two shelters with the same demands. Besides, the study shows that the four shelters have a more smooth evacuation rate compared to the two shelters.

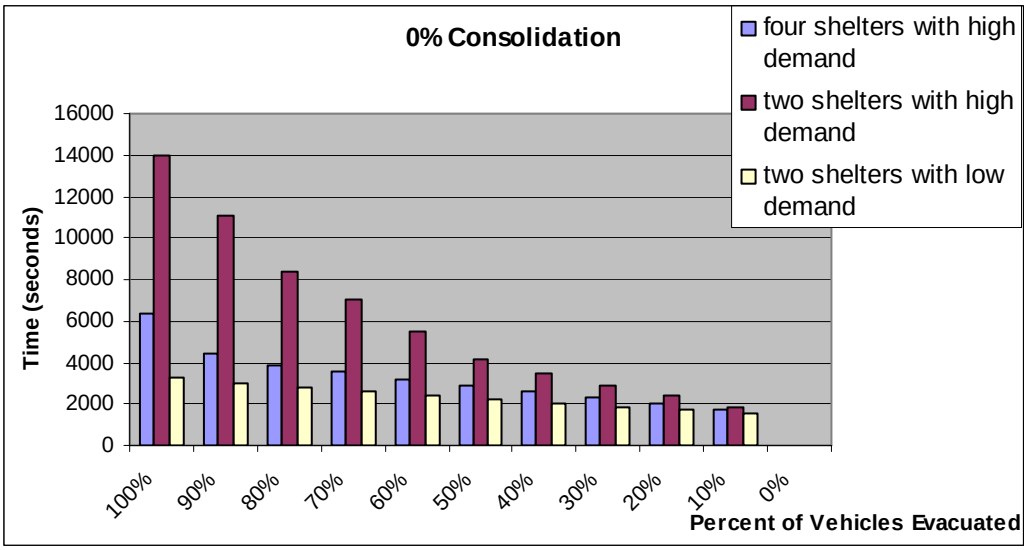


Figure 7-8. Comparison of percent of vehicles evacuated with different number of shelters under 0% consolidation scenario

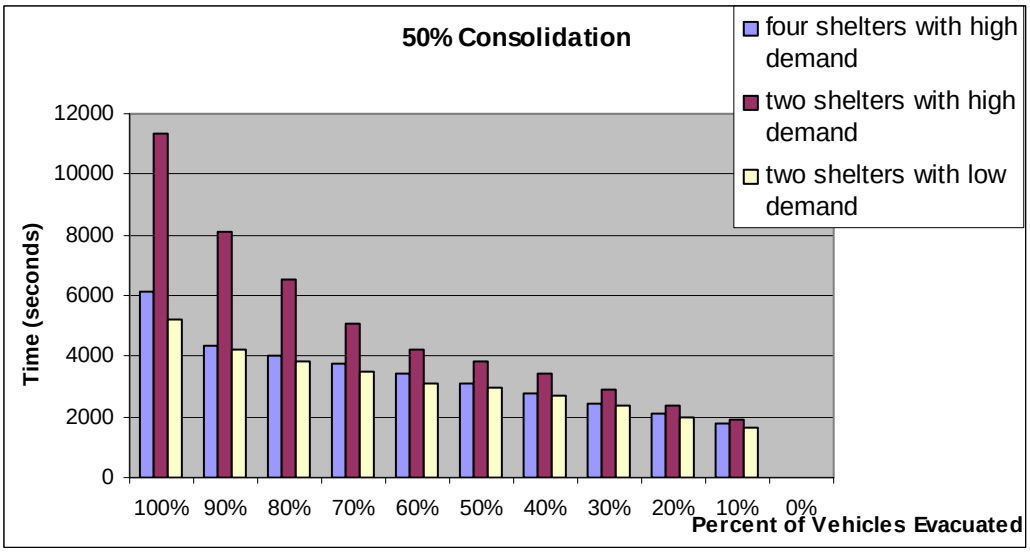


Figure 7-9. Comparison of percent of vehicles evacuated with different number of shelters under 50% consolidation scenario

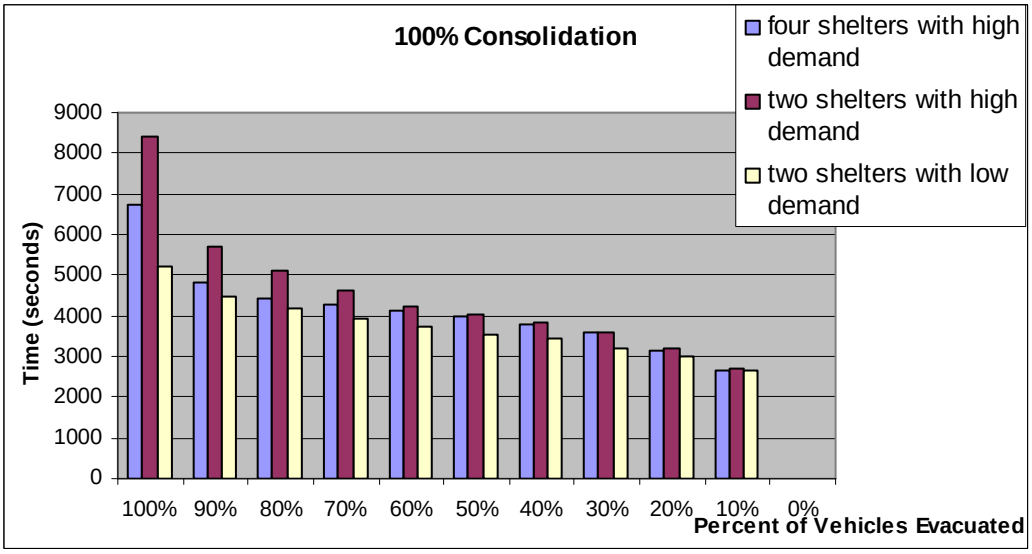


Figure 7-10. Comparison of percent of vehicles evacuated with different number of shelters under 100% consolidation scenario

Chapter 8: Summary and Conclusion

The primary goal of this dissertation is to better understand the collective behavior of a population during an emergency evacuation, and more specifically, the effect that certain family's intent to consolidate and evacuate together can have on overall evacuation performance metrics. This kind of consolidation process is usually ignored in the academic literature. Efforts have been made in this dissertation to investigate the problems and to build a new tool to model people's evacuation behavior, including the household consolidation behavior. Other critical issues related to the evacuation, such as information dissemination delays and evacuation awareness, and the network demographics and geography, have also been investigated.

The reminder of this chapter is organized as follows. The first section presents a summary of the research findings. The second section discusses the future research directions.

8.1 Summary of Research Findings

This dissertation includes a through review of the literature in this area. Chapter 2 includes an overview of the evacuation literature, important features and limitations of current evacuation models, and the existing observations of evacuation behavior patterns.

Extensive research has been conducted to study emergency evacuation preparedness and response processes. Some of them have concluded that household consolidation is an

important issue for certain types of evacuations. However, as shown in that chapter, although there are a large number of studies that have been conducted to measure traffic engineering impacts of evacuations, the specific social behaviors of household consolidation has not yet been fully explored, and more specifically, there is no reported work that has assessed the influence of evacuees' household consolidation behavior on various traffic conditions at a microscopic level.

Chapter 3 presents an approach to develop a simulation-based tool to study emergency evacuation that includes household evacuation behaviors. In this chapter, an Application Programming Interface (API) is written to track multi-class vehicles' household behaviors in both typical commuting traffic and emergency evacuation.

This chapter introduces the modeling framework, the development of a hierarchical data structure, and the initialization and sorting of the data structure. Most important, it introduces how the API tool deals with the vehicles in typical commuting traffic, and how the tool model the behaviors of multi-class drivers who are in a number of different states on the network respectively, such as yet to be released vehicles, en route vehicle, vehicles that have already arrived, and vehicles in the consolidation process.

In the Chapter 4, a primary application of the tool developed in chapter 3 is provided. It investigates the consolidation by household during the evacuation in both low demand and high demand situation. It shows that, with heavy demands, low consolidation rates seem to produce the longest evacuation times which may at first seem counter-intuitive.

This is an important result, because it suggests that it may actually be detrimental to have a very high percentage of vehicles attempting an outbound movement simultaneously, because they over-congest the network. This is not necessarily a controllable outcome – people will presumably return home and possibly consolidate regardless of instructions from the government – but it does shed considerable light on what the most useful and efficient traffic management strategies for evacuations might be, like staged evacuation.

Chapter 5 investigates the traffic volumes entering and leaving the network, as family consolidation is believed to have an impact on such traffic flow patterns. It indicates that high consolidation rates could delay the turning point to reverse the inbound lanes to outbound. This is an important result, because it shows that inappropriate implementation of contraflow strategy without considering people's household consolidation behavior may cause the network congestion to be detrimental during the evacuation. This chapter also reveals that if consolidation as family unit takes precedence, there is a substantial portion of inbound traffic at the beginning of the evacuation. Therefore, a proper sequence of reversing lanes, if staged evacuation is considered, is from the outer side of the network to the inner side. In this way, it can help reduce the bottleneck and gridlock in the network that caused by the household consolidation flows in the period immediately following a disaster.

Chapter 6 examines the evacuation dynamics with household consolidation under extended considerations, which include the efficiency of evacuation information dissemination, and the preparedness time the evacuees have for the evacuation. The

results show that consideration of information delays has a significant impact on the network performance. Besides, in combination with the preparedness time that evacuees may have, it shows that it is more realistic to include the communications between family members so that the total delay time could be decreased a lot.

Chapter 7 discusses that some demographics and geography factors that might affect household consolidation behaviors during an evacuation, such as the number of vehicles in a family and the number of shelters in the network. It shows that these factors have an impact on the evacuation performance in the incorporation with the evacuees' household social behaviors. Moreover, the number of shelters plays a more critical role. The increase of the number of shelters can significantly decrease the congestions in the network, and produces a more smooth evacuation rate.

8.2 Future Research

The work conducted in this dissertation leads to three possible future directions. First, other road networks with real demographic, geographical, and archived evacuation information need to be investigated. In this dissertation, we only examined a virtual network and provided a starting point. More experiments with some real networks should be conducted in order to examine whether the research findings from this dissertation are transferable to other networks.

Second, there are some interesting specific studies that can be performed with the proposed approach and tool from this work. For example, if there is an emergency event, like a chemical leak or man-made disaster, how critical components affect the performance of the evacuation, such as the location of the incident, its distance from the city center, the exit location, the locations of residential centers, and the percentage of consolidation.

Third, a more thorough investigation of contra-flow operations could be conducted for a specific network and demand distribution.

Appendix A: Paramics API Code

```
#define QPV3_TYPES

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>

#include "programmer.h"

#include "plugin_p.h"

#include "Data_Structr.h"

static float g_Evacuation_Start_Time = 1200;
static float g_WarmUp_Time = 18;

static VEHICLE * testvehicle;
static int g_altcolor = 1;
static int g_firstVehicle = 0;
static int g_trackVehicle = 20002;

static int g_nFamilies = 10000;
static int g_nNewTripDummyVehs = 1000;
static int g_nEvacDest = 4;
static int g_SimulationTime = 1440;
static int g_nTotalVehs = 100000;
static int g_nEnRouteVehs = 0;
static float g_max_Time=1000000;
static float g_max_Time_2=999999; //initilizae "stay-at-home" vehicle
static float g_max_Time_3=2000000;
static float g_WorkToHomeTrip_Delay = 28800.00; //This vehicle's departure time is 8-hr from
now
static int g_BoundaryZoneVeh = 20;
static float g_time_step=0;
static float g_time_step1=0;
static float g_agg = 300; //g_agg,Aggregation time interval is 300
static int g_ttVehCounts[10000]; //total link counts in a time interval for output purpose
static int g_tt[10000]; //total link counts in the planning horizon for output purpose
static int g_boundaryZone_North = 613;
static int g_boundaryZone_West = 614;
static int g_boundaryZone_South = 615;
static int g_boundaryZone_East = 616;
static int g_dummy_shelter = 700; //set a location (zone 700) for a dummy shelter (for yet to be
released vehicle to change destination later when en route)
static int g_dummy_home = 750;     //set a location (zone 750) for a dummy home (for stay at
home vehicle's destination)
static int g_dummyDest = 800; //set a location (zone 800) for a dummy vehicle used for
generating new trip during a simulation
static int g_ArrivedShelterVehicle = 0;
static Bool evacuation_started=FALSE;
static Bool ytb_Veh=FALSE, enRoute_Veh=FALSE;
```

```
struct Enroute_Vehicle *enroute_vehHeadPtr=NULL;
struct Enroute_Vehicle *enroute_vehTailPtr=NULL;


//Pointer always pointing to the head of the veh link list
struct Vehicle *g_VehList_headPointer=NULL;
//Pointer to the current veh ready to be released to the network
struct Vehicle *g_VehPointer=NULL;
struct Vehicle *g_LastVehPointer=NULL;
struct Vehicle *g_DummyVehPointer=NULL; //A Pointer always pointing to the head of the dummy veh
struct Vehicle *g_LastDummyVehPointer=NULL; // A pointer always pointng to the previous head of dummy veh list (initial state: pointing to the tail of the undummy veh)

struct Vehicle *g_NewTripVehPointer=NULL;
struct Vehicle *g_LastNewTripVehPtr=NULL;


FILE* g_Results;
FILE* g_Results_2;
FILE* g_Results_3;
FILE* g_Results_4;
FILE* g_Results_flow;
char fname[_MAX_PATH],fname1[_MAX_PATH];;
char fname_flow[_MAX_PATH], fname_YTB[_MAX_PATH];
int g_totalLinks;
static Bool scenarioType[40];
static char *scenarioName[2]={"scenario 1","scenario 2"};


NET_USERDATA *netdata = NULL;

/* Function Prototypes */
static void pp_NormalRandomNum(float *, float *);
//static void pp_insert_dummy_veh(VEHICLE*, VEHICLE*, VEHICLE*);
static void pp_init_data(void);
static void pp_sort_data(void);
static void pp_print_data(void);
static long pp_Bernoulli(double);
static long LocationGenerator(int, int);
static int VehicleGenerator(void);
static int VehTypeGenerator(void);
static int VehicleCapacityGenerator(void);
static int HomeboundGenerator(void);
static void pp_start_time_assign(int, float*);
static void pp_release_vehicle(int);
static float pp_Uniform(float, float);
static int pp_consolidated_dest(int);
static int VehicleODGenerator(void);
static double pp_triangular(double, double,double);
static void         pp_draw_yet_to_be_released_vehicles(void);

struct Vehicle
{
```

```
	int WorkLocation;
	int HomeLocation;
	int orgZone;
	int desZone;
	int capacity;
	struct Vehicle* next;
	float StartTime;
	struct Family* fm_next;
	float DelayTime;
	int vehType;
	int VehicleID;
	int NewDest;
	float ChangeTime;
	float AwareTime;
};

struct Family
{
	int NumOfVehs;
	int NumOfArrivedVehs;
	int HomeLocation;
	int HomeBound; //1-homebound (consolidated); 0-non-homebound (non-consolidated)
	int FamilyID;
	struct Vehicle* VehicleList;
	struct Family* next;
};

typedef struct NET_USERDATA_s
{
  int NumOfFamilies;
  struct Family* FamilyList;
} NET_USERDATA;

void qpx_DRW_modelView(void)
{
		qps_DRW_solid();
		qps_DRW_highlightLink(link, API_RED);
		qps_DRW_moveToVehicleHome(testvehicle);
		qps_DRW_hollowCircleXY(0, 0, 10);
		qps_DRW_vehicleTag(testvehicle, API_RED, 1, 3, scenarioName[1]);
		qps_DRW_forceVisibleObjectsRebuild(TRUE);
		qps_DRW_forceTimeStepRedraw(TRUE);

}

/* ---------------------------------------------------------------------
 * call qpx_NET_postOpen once when the full network has been read into modeller
 * --------------------------------------------------------------------- */
void qpx_NET_postOpen(void)
{

	int zone = 0;
	int iLinkIdex;


	qps_DRW_forceTimeStepRedraw(TRUE);
```

```
    /* open the output file */
        sprintf(fname1,"c:/Arrived_Veh.dat");
        g_Results=fopen(fname1,"w");
        fprintf(g_Results,"Origin  Dest  HomeLoc  WorkLoc  Homebound  StartTime\n");
        fclose(g_Results);
    /****End open the file*********************/


    //Open the link performance file, comment it out when not required
    sprintf(fname_flow,"c:/LP_Flow.dat");
    g_Results_flow=fopen(fname_flow,"w");
    fprintf(g_Results_flow,"Time      Link_ID      UpNode      DownNode      Flow\n");
    fclose(g_Results_flow);

        g_totalLinks=qpg_NET_links();
        for(iLinkIdex=0; iLinkIdex<g_totalLinks;iLinkIdex++)
        {
            g_tt[iLinkIdex]=0;
        }
    /*****************************************/

        g_time_step1=0.5;
        qps_CFG_timeStep(g_time_step1);

        g_time_step=qpg_CFG_timeStep();
        g_time_step1=qpg_CFG_timeStepDetail();

        qps_GUI_printf("Code initiated\n");

        /* call to initialize the data structures for Network, Family and Vehicle*/
        pp_init_data();

    /* call to sort the veh list according to the start time */
        pp_sort_data();

        /* call to print out the vehicle information after initilization and double sorting*/
        pp_print_data();

        /* Point to the head of the veh linked list */
        g_VehPointer=netdata->FamilyList->VehicleList;

        /* Pointer always point to the head of the veh linked list (head pointer)*/
        g_VehList_headPointer=netdata->FamilyList->VehicleList;

}


//This function is called for each zone in the network once per simulation time step.
void qpx_ZNE_timeStep(ZONE* zone)
{
        float current_time, tempStart_Time;
        float delay_a=0, delay_b=900;
        int dest_zone;
        int veh_id1=0, veh_id2=0;
        int CurrentOriginZone, iidebug;
```

```
	Bool cont=TRUE;
	struct Vehicle * veh_temp= NULL;
	VEHICLE* vehicle_S1=NULL;
	float test_x, test_y, test_z;

	current_time = qpg_CFG_simulationTime();
	CurrentOriginZone = qpg_ZNE_index(zone);  //This function returns the network wide index for the specified zone.

	//This part is used to generate the visualization
	if(CurrentOriginZone == 1)
	{
		qpg_POS_crossHair(&test_x, &test_y, &test_z);
		qps_GUI_printf("crossHair x=%5.2f, y=%5.2f, z=%5.2f\n", test_x, test_y, test_z);
		qpg_POS_viewPoint(&test_x, &test_y, &test_z);
		qps_GUI_printf("viewPoint x=%5.2f, y=%5.2f, z=%5.2f\n", test_x, test_y, test_z);

	}


//  release a vehicle from a double-sorted (by zone and time) vehicle list
   while((g_VehPointer!=NULL) && cont)
	{
		if(g_VehPointer->StartTime<=current_time) //If the current veh start time less than the current simulation time, release
		{
	       if(g_VehPointer->orgZone ==CurrentOriginZone)
			{
				dest_zone = g_VehPointer->desZone;
			//Debugging
			//	if(g_VehPointer->desZone == 616 && g_VehPointer->orgZone ==160) //&& qpg_VHC_startTime(vehicle)==2445.5)
			//		iidebug=1;
			  //End Debugging

		if(g_VehPointer->vehType == 1){  //This is a type of at-home vehicle which won't be released when evacuation starts
					//do nothing
				}
				else
				{
					pp_release_vehicle(dest_zone);
				}

				//qps_GUI_printf("Veh scheduled to be released from Org %d to Dest %d, at departure time : %5.2f\n", g_VehPointer->WorkLocation, g_VehPointer->HomeLocation, g_VehPointer->StartTime);
				g_LastVehPointer = g_VehPointer;
				g_VehPointer=g_VehPointer->next;
			}
			else
				cont = FALSE;
		}
		else
			cont = FALSE;
	}
```

```
// yet-to-be-released vehicle's simulation
//GO over entire link list at Zone 1 (the first zone) change the departure time, destination zone of
//the entire veh link list at zone 1 (the first zone)

        veh_temp=g_VehPointer;
        if(current_time>=g_Evacuation_Start_Time-0.05                                    &&
current_time<=g_Evacuation_Start_Time+0.05 && ytb_Veh==FALSE)
        {
      ytb_Veh=TRUE;

        // This is used to create a report for checking yet-to-be-released vehicles. It should be
comment out when necessary
        sprintf(fname_YTB,"c:/YTB_Released_Veh.dat");
        g_Results_3=fopen(fname_YTB,"w");


      while(veh_temp!=NULL&&veh_temp->StartTime!=g_max_Time)
                {
                        dest_zone = veh_temp->desZone;

          if (dest_zone == veh_temp->HomeLocation)
                        {
            if(veh_temp->fm_next->HomeBound == 1) //is this veh a consolidated one?
                         {
                                if(veh_temp->StartTime        <        (current_time+veh_temp-
>DelayTime))
                                { //is this vehicle's planned trip earlier than evacaution time plus
delay?
                                        veh_temp->ChangeTime = current_time + veh_temp-
>DelayTime;     //set changetime to evacuation time + delay
                                        veh_temp->NewDest = veh_temp->HomeLocation;
                                        veh_temp->AwareTime    =    veh_temp->ChangeTime;
//add on 6/12/09
                                }
                                else
                                {
                                        veh_temp->StartTime=current_time     +     veh_temp-
>DelayTime;  //set release time to evacuation time + delay
                                        veh_temp->AwareTime = veh_temp->StartTime;  //add
on 6/12/09
                                }
                        }
                        else if(veh_temp->fm_next->HomeBound == 0) //this vehicle is not a
consolidated one
                        {
                                 if(veh_temp->StartTime        <        (current_time+veh_temp-
>DelayTime))
                                 {
                                        veh_temp->NewDest = g_dummy_shelter;
                                        veh_temp->ChangeTime = current_time + veh_temp-
>DelayTime;
                                        veh_temp->AwareTime    =    veh_temp->ChangeTime;
//add on 6/12/09
                                 }
                                 else
```

```
                                        {
                                                veh_temp->StartTime=current_time     +     veh_temp-
>DelayTime;
                                                veh_temp->desZone                                    =
pp_consolidated_dest(dest_zone); //Set its dest to the closest shelter
                                                veh_temp->AwareTime = veh_temp->StartTime;  //add
on 6/12/09
                                        }
                                  }
                                }
                                else //destination != home
                                {
                  if(veh_temp->fm_next->HomeBound == 1)  //is this veh a consolidated one?
                                  {
                                        if(veh_temp->orgZone == veh_temp->HomeLocation) //origin =
home
                                        {
                                                if(veh_temp->StartTime    <    (current_time+veh_temp-
>DelayTime))
                                                {
                                                        veh_temp->NewDest          =          veh_temp-
>HomeLocation;
                                                        veh_temp->ChangeTime    =    current_time    +
veh_temp->DelayTime;
                                                        veh_temp->AwareTime        =        veh_temp-
>ChangeTime; //add on 6/12/09
                                                }
                                                else
                                                {
                                                        veh_temp->vehType = 1;  //vehType 1 --- vehicle
at home when evacuation starts; (prevent it from being released)
                                                        veh_temp->fm_next->NumOfArrivedVehs++;
//this veh is already at home, increase the number of arrived vehicles by one
                                                        veh_temp->AwareTime    =    current_time    +
veh_temp->DelayTime;  //add on 6/12/09
                                                }
                                        }
                                        else  //origin != home
                                        {
                                                if(veh_temp->StartTime    <    (current_time+veh_temp-
>DelayTime)) {
                                                        veh_temp->NewDest          =          veh_temp-
>HomeLocation;
                                                        veh_temp->ChangeTime=current_time          +
veh_temp->DelayTime;
                                                        veh_temp->AwareTime        =        veh_temp-
>ChangeTime; //add on 6/12/09
                                                }
                                                else
                                                {
                                                        veh_temp->StartTime=current_time              +
veh_temp->DelayTime;
                                                        veh_temp->desZone=veh_temp-
>HomeLocation;
                                                        veh_temp->AwareTime = veh_temp->StartTime;
//add on 6/12/09
```

```
                                        }
                                    }
                                }
                                else if(veh_temp->fm_next->HomeBound == 0)//this vehilce is not a consolidated one
                                {
                            if(veh_temp->StartTime < (current_time+veh_temp->DelayTime)) {
                                        veh_temp->NewDest = g_dummy_shelter;
                                        veh_temp->ChangeTime = current_time + veh_temp->DelayTime;
                                        veh_temp->AwareTime = veh_temp->ChangeTime; //add on 6/12/09
                                    }
                                    else
                                    {
                                        veh_temp->StartTime=current_time + veh_temp->DelayTime;
                                        veh_temp->desZone = pp_consolidated_dest(dest_zone); //Set its dest to the closest shelter
                                        veh_temp->AwareTime = veh_temp->StartTime; //add on 6/12/09
                                    }
                                }
                            } //else Destination != home

                    //This is used to create a report for checking yet-to-be-released vehicles. It should be comment out when necessary
                    fprintf(g_Results_3,"Vehicle %d release time %7.2f, origin %d, dest %d, home %d, work %d, FamilyID %d, homebound %d, vehType %d, newdest %d, changeTime %7.2f, delay %7.2f\n", veh_temp->VehicleID, veh_temp->StartTime, veh_temp->orgZone, veh_temp->desZone, veh_temp->HomeLocation, veh_temp->WorkLocation, veh_temp->fm_next->FamilyID, veh_temp->fm_next->HomeBound, veh_temp->vehType, veh_temp->NewDest, veh_temp->ChangeTime, veh_temp->DelayTime);

                        //move pointer to point to next
                        veh_temp=veh_temp->next;
                    } //end while(veh_temp!=NULL&&veh_temp->StartTime!=g_max_Time)

                    //Re-sort Vehicles
                    pp_sort_data();

                    // This is used to create a report for checking yet-to-be-released vehicles. It should be comment out when necessary
                    fclose(g_Results_3); //generate an output


                    //pp_draw_yet_to_be_released_vehicles();

            } //end if(current_time>=g_Evacuation_Start_Time-0.05...)
}

//This function is called once at the start of each time step of simulation time
//Report Network-wide Link Statistical Results
void qpx_NET_timeStep()
{
            float ttCounter=0;
```

```
	int iLinkIdex,jLinkIdex,ttLanes;
	LINK* link_tmp;
	int VehID = 0;
	int currentZonetemp = 0;
	Bool cont=TRUE;

	struct Family *tempFamilyPtr;
	struct Vehicle *tempVehiclePtr;

	// Output link performance

	ttCounter=qpg_CFG_simulationTime();

	// Count the total number of vehicles in the network
	for(iLinkIdex=0; iLinkIdex<g_totalLinks;iLinkIdex++){
	  link_tmp=qpg_NET_linkByIndex(iLinkIdex+1);  //This function returns a pointer to the link with the specified network wide index
	  ttLanes=qpg_LNK_lanes(link_tmp); //This function returns the number of lane on the link.
	  for(jLinkIdex=0;jLinkIdex<ttLanes; jLinkIdex++) {
	    g_ttVehCounts[iLinkIdex]+=qpg_LNK_vehicles(link_tmp, jLinkIdex);  //This function returns the number of vehicles in the given lane on the specified link.
	  }
	}

	//Output link performance for each aggregated time interval (300 seconds in this case)
	if(fmod(ttCounter,g_agg)==0) //This function returns the remainder of dividing the arguments.
	{
	  g_Results_flow=fopen(fname_flow,"a");
      fprintf(g_Results_flow, "%5.2f\n",qpg_CFG_simulationTime());
	  for(iLinkIdex=0; iLinkIdex<g_totalLinks;iLinkIdex++)
	  {
		//fprintf(g_Results_flow,"%5.2f  %7d  %8d \n", qpg_CFG_simulationTime(), iLinkIdex, g_ttVehCounts[iLinkIdex]);
		link_tmp=qpg_NET_linkByIndex(iLinkIdex+1);

		fprintf(g_Results_flow,"%5.2f  %7d  %7d  %7d  %8d \n", qpg_CFG_simulationTime(), \
		 iLinkIdex,  qpg_NDE_index(qpg_LNK_nodeStart(link_tmp)), qpg_NDE_index(qpg_LNK_nodeEnd(link_tmp)), g_ttVehCounts[iLinkIdex]);
		g_tt[iLinkIdex]+=g_ttVehCounts[iLinkIdex];
		g_ttVehCounts[iLinkIdex]=0;
	  }
	  fclose(g_Results_flow);
	}

}


//This function is called for each vehicle in the network, for each link in the network, once per simulation time step.
void qpx_LNK_vehicleTimeStep(LINK* link, VEHICLE* vehicle)
{
	float ttCounter=0, Vehstart_time=0;
	float delay_a=0, delay_b=60;
	int currentZonetemp=0, randomZone=0, iidebug=0;
```

```
	Bool cont=TRUE;
	struct Vehicle *tempVehiclePtr;
	struct Enroute_Vehicle *enroute_veh;

	float test_x, test_y, test_z, test_b, test_g;

	ttCounter=qpg_CFG_simulationTime();
	currentZonetemp = qpg_LNK_zone(link); //This function returns the index of the zone associated with the link.

//	qps_GUI_printf("Vehicle's origin %d, destination %d, start time %d \n", qpg_VHC_origin(vehicle), qpg_VHC_destination(vehicle), qpg_VHC_startTime(vehicle));

	//debug
	//for the first vehicle appears at the network, change its color and pass its pointer to "testvehicle" pointer
	//Draw Vehicle part, comment out when necessary

   tempVehiclePtr=(struct Vehicle*)qpg_VHC_userdata(vehicle);

//This part is used to generate the visualization

	if(tempVehiclePtr->VehicleID == g_firstVehicle)
	{
		qps_DRW_forceTimeStepRedraw(TRUE);
		qpg_POS_vehicle(vehicle, link, &test_x, &test_y, &test_z, &test_b, &test_g);
		qps_GUI_printf("vehicle x=%5.2f, y=%5.2f, z=%5.2f\n", test_x, test_y, test_z);

		//qps_POS_viewPoint(test_x,test_y,750);
		qps_POS_crossHair(-test_x+5601,test_y,0);

		if (g_altcolor == 1) {
			qps_DRW_vehicleColour(vehicle, API_RED);
		}
		else
		{
			qps_DRW_vehicleColour(vehicle, API_GREEN);
		}
		g_altcolor = 1-g_altcolor;
		testvehicle = vehicle;
	}


	if(tempVehiclePtr->VehicleID == g_trackVehicle)
	{
		qps_DRW_forceTimeStepRedraw(TRUE);
		qpg_POS_vehicle(vehicle, link, &test_x, &test_y, &test_z, &test_b, &test_g);
		qps_GUI_printf("vehicle x=%5.2f, y=%5.2f, z=%5.2f\n", test_x, test_y, test_z);

		//qps_POS_viewPoint(test_x,test_y,750);
		qps_POS_crossHair(-test_x+5601,test_y,0);
		if (g_altcolor == 1) {
			qps_DRW_vehicleColour(vehicle, API_RED);
		}
		else
		{
```

```
                qps_DRW_vehicleColour(vehicle, API_GREEN);
            }
            g_altcolor = 1-g_altcolor;
            testvehicle = vehicle;
        }



        //1. Check if the current time step equals to the evacuation start time, if so, do the following
        if(ttCounter==g_Evacuation_Start_Time )
        {
    enRoute_Veh=TRUE;


            //2. go through the Vehicles on the vechile linked list
    tempVehiclePtr=netdata->FamilyList->VehicleList;

            sprintf(fname_YTB,"c:/EnRoute_Veh.dat");
            g_Results_4=fopen(fname_YTB,"w");


            while (tempVehiclePtr != NULL) //&& cont)
            {
       Vehstart_time=(int)(tempVehiclePtr->StartTime+0.5);   //convert a float value of StartTime to an integer (by plus .5)

        //      tempVehiclePtr=(struct Vehicle*)qpg_VHC_userdata(vehicle);

                    if(qpg_VHC_startTime(vehicle)==Vehstart_time &&
qpg_VHC_origin(vehicle)==tempVehiclePtr->orgZone &&
qpg_VHC_destination(vehicle)==tempVehiclePtr->desZone) //three parameters to identify this vehicle in Paramics is the one we are looking for in our data structure
                    {
         cont=FALSE;

                      if(qpg_VHC_destination(vehicle)==tempVehiclePtr->HomeLocation) //Destination = Home?
                      {
          //Check if the vehicle is a consolidated vehicle
            if(tempVehiclePtr->fm_next->HomeBound == 1) //this vehicle is a consolidated vehicle
                                {
                                    //This Vehicle is a consolidated vehicle heading towards its home (consolidated point), do nothing
                                    tempVehiclePtr->AwareTime = ttCounter + tempVehiclePtr->DelayTime; //add on 6/12/09
                                    g_nEnRouteVehs++;  //add on 6/12/09

                                }
                            else
                                {
                                    //This vehicle will go to the dummy shelter/exit
              // tempVehiclePtr->NewDest = pp_consolidated_dest(currentZonetemp); //Set its dest to the closest shelter
                                    tempVehiclePtr->NewDest = g_dummy_shelter;
```

```
                                        tempVehiclePtr->ChangeTime = ttCounter +
tempVehiclePtr->DelayTime;
                                        tempVehiclePtr->AwareTime = tempVehiclePtr-
>ChangeTime; //add on 6/12/09
                                        qps_DRW_vehicleColour(vehicle, API_RED);

                                        g_nEnRouteVehs++;
                                }
                        }
                        else  //dest!=home
                        {
                                if(qpg_VHC_destination(vehicle)==tempVehiclePtr-
>WorkLocation)
                                {
                                        if(tempVehiclePtr->fm_next->HomeBound == 1) {
                                                //change the en route vehicle's destination to
its home
                                                tempVehiclePtr->NewDest = tempVehiclePtr-
>HomeLocation;
                                                tempVehiclePtr->ChangeTime = ttCounter +
tempVehiclePtr->DelayTime;
                                                tempVehiclePtr->AwareTime =
tempVehiclePtr->ChangeTime; //add on 6/12/09
                                                qps_DRW_vehicleColour(vehicle, API_RED);
                                                g_nEnRouteVehs++;
                                        }
                                        else {
                                                //Change the en-route vehicle's destination to
the dummy shelter
                                                tempVehiclePtr->NewDest =
g_dummy_shelter;
                                           tempVehiclePtr->ChangeTime = ttCounter +
tempVehiclePtr->DelayTime;
                                                tempVehiclePtr->AwareTime =
tempVehiclePtr->ChangeTime; //add on 6/12/09
                                             qps_DRW_vehicleColour(vehicle, API_RED);
                                                g_nEnRouteVehs++;
                                        }
                                }//end         if(qpg_VHC_destination(vehicle)==tempVehiclePtr-
>WorkLocation)
                                else //this is a meander vehicle
                                {
                                        if(tempVehiclePtr->fm_next->HomeBound == 1)
                                        {
                                                //Change the en-route vehicle's destination to
its home
                                                tempVehiclePtr->NewDest=tempVehiclePtr-
>HomeLocation;
                                                tempVehiclePtr->ChangeTime = ttCounter +
tempVehiclePtr->DelayTime;
                                                tempVehiclePtr->AwareTime =
tempVehiclePtr->ChangeTime; //add on 6/12/09
                                                qps_DRW_vehicleColour(vehicle, API_RED);
                                                g_nEnRouteVehs++;
                                }
                                        else
```

```
                                        {
                                                //Generate a "meanderer" vehicle with random
destination
                                                //do {
                                                //    randomZone = LocationGenerator(17,18);
//Generate random location for this vehicle
                                                //} while (randomZone == tempVehiclePtr-
>desZone);
                                                //tempVehiclePtr->NewDest=randomZone;
                                                tempVehiclePtr->NewDest=g_dummy_shelter;
                                                tempVehiclePtr->ChangeTime = ttCounter +
tempVehiclePtr->DelayTime;
                                                tempVehiclePtr->AwareTime =
tempVehiclePtr->ChangeTime; //add on 6/12/09
                                                qps_DRW_vehicleColour(vehicle, API_RED);
                                                g_nEnRouteVehs++;
                                }
                                        }//else       if(qpg_VHC_destination(vehicle)==tempVehiclePtr-
>WorkLocation)

                                }//else                (qpg_VHC_destination(vehicle)==tempVehiclePtr-
>HomeLocation)

                        }//if             (qpg_VHC_startTime(vehicle)==Vehstart_time          &&
qpg_VHC_origin(vehicle)==tempVehi...

            //fprintf(g_Results_4,"At time %7.2f, Vehicle %d release time %7.2f, origin %d, dest %d,
home %d, work %d, FamilyID %d, homebound %d, vehType %d, newdest %d,
changeTime %7.2f, delay %7.2f\n", ttCounter, tempVehiclePtr->VehicleID, tempVehiclePtr-
>StartTime, tempVehiclePtr->orgZone, tempVehiclePtr->desZone, tempVehiclePtr-
>HomeLocation, tempVehiclePtr->WorkLocation, tempVehiclePtr->fm_next->FamilyID,
tempVehiclePtr->fm_next->HomeBound, tempVehiclePtr->vehType, tempVehiclePtr->NewDest,
tempVehiclePtr->ChangeTime, tempVehiclePtr->DelayTime);

                        //move the pointer to the next vehicle in the linked list
                        tempVehiclePtr = tempVehiclePtr->next;

                } //while(...

                fclose(g_Results_4); //generate an output


        }//if(ttCounter>=g_Evacuation_Start_Time-
0.05&&ttCounter<=g_Evacuation_Start_Time+0.05

//Re-route enroute vehs to the new destinations (shelters) after the evacuation starts
//Create Enroute veh link list--4/11/2009
//        if(g_nEnRouteVehs>0 && enRoute_Veh==TRUE && ttCounter>g_Evacuation_Start_Time)
        if(enRoute_Veh==TRUE && ttCounter>g_Evacuation_Start_Time)
        {
    tempVehiclePtr=(struct Vehicle*)qpg_VHC_userdata(vehicle);
                if(tempVehiclePtr->ChangeTime     >=     ttCounter-0.05     ||     tempVehiclePtr-
>ChangeTime <= ttCounter+0.05)
                {
                        if(tempVehiclePtr->NewDest == g_dummy_shelter){
                                //Change the current vehicle's destination to the nearest shelter
```

```
a new trip from current location to the nearest shelter
                                        tempVehiclePtr->desZone                                         =
pp_consolidated_dest(currentZonetemp);
                                        qps_VHC_destination(vehicle, tempVehiclePtr->desZone, 0);
                                        tempVehiclePtr->ChangeTime = 0;
            g_nEnRouteVehs--;
                                        qps_DRW_vehicleColour(vehicle, API_BLUE);
                                }
                                else if(tempVehiclePtr->NewDest == tempVehiclePtr->HomeLocation){
                                        //Change the current vehicle's destination to its consolidated
point (home)
                                        qps_VHC_destination(vehicle,    tempVehiclePtr->HomeLocation,
0);
                                        tempVehiclePtr->ChangeTime = 0;
            g_nEnRouteVehs--;
                                        qps_DRW_vehicleColour(vehicle, API_BLUE);
                                }
                        }
                }
}


/*Vehicle user data structure set function*/
void qpx_VHC_release(VEHICLE* vehicle) /*This function is called when a vehicle is released
from a zone.*/
{
        qps_VHC_userdata(vehicle, (VHC_USERDATA*) g_LastVehPointer); //set the user data
structure associated with the specified vehicle.

        if (g_firstVehicle == 0 )
        {
                g_firstVehicle = g_LastVehPointer->VehicleID;  //first vehicle to be released

                //Debug
           qps_GUI_printf("First vehicle released has ID number %d.", g_firstVehicle);

        }


        //debug
//      tempVehiclePtr=(struct Vehicle*)qpg_VHC_userdata(vehicle);

        if(g_LastVehPointer->VehicleID == 20002)
        {
                g_trackVehicle = g_LastVehPointer->VehicleID;  //the test vehicle to be released

                //Debug
             qps_GUI_printf("Test vehicle released has ID number %d and dest zone %d.\n",
g_trackVehicle, qpg_VHC_destination(vehicle));
        }
        //end debug



        //Debug
        //qps_GUI_printf("Veh released from Org %d to Dest %d, at departure time : %5.2f\n",
```

```
qpg_VHC_origin(vehicle), qpg_VHC_destination(vehicle), qpg_CFG_simulationTime());


        //Generate a output for currently en-route vehicles
        sprintf(fname,"c:/Veh_Release.dat");
   g_Results=fopen(fname,"w");
        fprintf(g_Results,"Origin  Dest  HomeLoc WorkLoc Homebound StartTime\n");
        fprintf(g_Results,"Veh released from Org %d to Dest %d, at departure time : %5.2f\n",
qpg_VHC_origin(vehicle), qpg_VHC_destination(vehicle), qpg_CFG_simulationTime());
        fclose(g_Results);

}

/*Vehicles (that have already arrived)' Simulation & Vehicle Consolidation Functions are called
here*/
void qpx_VHC_arrive(VEHICLE* vehicle, LINK* link, ZONE* zone)  //This function is called when
a vehicle arrives at its destination
{
        float arrival_time, current_time, r;
        int num_vhc_arrive=0;
        //int icounter=0;
        int icounter=1; //add by Ke at 7/15/09
        int iidebug;
        int destIndex=0, currentZonetemp=0;
        struct Vehicle *VehArrive;
        struct Vehicle *CurrentVeh_temp=NULL;
        struct Vehicle *CurrentVeh_temp_front=NULL;
        VHC_USERDATA *user_veh=NULL;
        struct Family *FamArrive;
        float delay_start_time = 0;
        float delay_end_time = 1800.0;
        //float delay_end_time = 300.0;
        float maxTime=0.0, minTime=0.0;
        int originzone,destinationzone;
        BOOL cont=FALSE, cont0=0, cont_trip=0;
        struct Vehicle *VehPointer;
        int totalNumbofVeh;
        int ConsolidatedDestination, curr_home_location, randomZone;

        VehArrive=(struct Vehicle*) qpg_VHC_userdata(vehicle); //return the user data structure
associated with the vehicle

        current_time = qpg_CFG_simulationTime();//Determine Current time
   destIndex=qpg_VHC_destination(vehicle); //This function returns an index to the vehicles
destination zone.
        currentZonetemp = qpg_LNK_zone(link);//This function returns the index of the zone
associated with the link.

/
        //Generate a output for arrived vehicles
      g_Results=fopen(fname1,"a"); //add 7/13/09
        fprintf(g_Results,"Origin  Dest  HomeLoc WorkLoc Homebound StartTime\n");
        fprintf(g_Results,"%d  %d  %d  %d  %d  %7.2f \n", VehArrive->orgZone,
VehArrive->desZone, VehArrive->HomeLocation, VehArrive->WorkLocation, VehArrive->fm_next-
>HomeBound, VehArrive->StartTime); //add 7/13/09
        fclose(g_Results);  //add 7/13/09
```

```
	if(current_time<g_Evacuation_Start_Time-0.05) //Evacuation didn't start yet
	{
	  if(destIndex==VehArrive->WorkLocation) //Dest = work
	  {
			//Generate a new work-to-home trip with depature time 8 hrs from now
			if(g_NewTripVehPointer == NULL)
			{
				qps_GUI_printf("No new trip is ready to be generated for this veh!!!\n");
			}
			else
			{
				g_NewTripVehPointer->StartTime                                          =
current_time+g_WorkToHomeTrip_Delay; //This vehicle's departure time is 8-hr from now
				g_NewTripVehPointer->orgZone = currentZonetemp;
				g_NewTripVehPointer->desZone = VehArrive->HomeLocation;
				g_NewTripVehPointer->DelayTime = 0.0;
				g_NewTripVehPointer->ChangeTime = 0.0;
				g_NewTripVehPointer->fm_next->HomeBound  =  VehArrive->fm_next-
>HomeBound;

				g_nEnRouteVehs--;
				qps_DRW_vehicleColour(vehicle, API_ORANGE);

				//Insert this new vehicle into the sorted vehicle linked list
					if(g_LastNewTripVehPtr!=NULL&&g_NewTripVehPointer!=NULL)
					{
						if((g_NewTripVehPointer-
>StartTime>g_LastNewTripVehPtr->StartTime))
						{
							cont_trip=1;
							if(g_NewTripVehPointer-
>next!=NULL&&g_NewTripVehPointer->next->StartTime==g_max_Time_3)
							{

	g_LastNewTripVehPtr=g_NewTripVehPointer;

	g_NewTripVehPointer=g_NewTripVehPointer->next;
							}
							else if (g_NewTripVehPointer->next==NULL)
							{

	g_LastNewTripVehPtr=g_NewTripVehPointer;
									g_NewTripVehPointer=NULL;

							}
						}
					}//end
if(g_LastNewTripVehPtr!=NULL&&g_NewTripVehPointer!=NULL)
					if(! cont0)
					{
						//Step 1: isolate the new generated veh from the dummy
vehs list

if(g_LastNewTripVehPtr!=NULL&&g_NewTripVehPointer!=NULL)
```

```
                                        {
                                                g_LastNewTripVehPtr-
>next=g_NewTripVehPointer->next;
                                        }
                                        //Step 2: Insert the isolated new generated veh to the
sorted veh link list
                                        if(g_NewTripVehPointer!=NULL)
                                        {
                                                if(g_NewTripVehPointer->StartTime
<g_VehPointer ->StartTime)
                                                { // insert front
                                                        g_NewTripVehPointer->next=
g_VehPointer;
                                                        g_VehPointer=g_NewTripVehPointer;
                                                }
                                                else                    if(g_NewTripVehPointer-
>StartTime==g_VehPointer ->StartTime)
                                                {
                                                        if(g_NewTripVehPointer-
>orgZone<=g_VehPointer->orgZone) //OrgZone?
                                                        {
                                                                g_NewTripVehPointer->next=
g_VehPointer;
        g_VehPointer=g_NewTripVehPointer;
                                                        }
                                                        else
                                                        {

        CurrentVeh_temp=g_VehPointer;

        CurrentVeh_temp_front=CurrentVeh_temp;
                                                                cont = TRUE;
                                                                while
((CurrentVeh_temp!=NULL) && cont)
                                                                {

        if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                                        {

        if(g_NewTripVehPointer->orgZone<=CurrentVeh_temp->orgZone)
                                                                                {

        g_NewTripVehPointer->next=CurrentVeh_temp;

        CurrentVeh_temp_front->next=g_NewTripVehPointer;
                                                                                        cont    =
FALSE;
                                                                                }
                                                                                else {

        CurrentVeh_temp_front=CurrentVeh_temp;

        CurrentVeh_temp=CurrentVeh_temp->next;
                                                                                }
                                                                        }
```

```
//if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                    else //insert the dummy veh to the last postion with the same starttime
                                                    {
        g_NewTripVehPointer->next=CurrentVeh_temp;
        CurrentVeh_temp_front->next=g_NewTripVehPointer;
                                                            cont = FALSE;
                                                    }
                                                }                   //While ((CurrentVeh_temp!=NULL) && cont)
                                            }               //if(g_NewTripVehPointer->orgZone<=g_VehPointer->orgZone)
                                        }//else                 if(g_NewTripVehPointer->StartTime==g_VehPointer ->StartTime)
                                        else                    if(g_NewTripVehPointer->StartTime>g_VehPointer ->StartTime)
                                        {
                                            CurrentVeh_temp=g_VehPointer;
        CurrentVeh_temp_front=CurrentVeh_temp;
                                            cont = TRUE;
                                            while ((CurrentVeh_temp!=NULL) && cont)
                                            {
                                                if(g_NewTripVehPointer->StartTime>CurrentVeh_temp->StartTime)
                                                {
        CurrentVeh_temp_front=CurrentVeh_temp;
        CurrentVeh_temp=CurrentVeh_temp->next;
                                                }
                                                else    if(g_NewTripVehPointer->StartTime<CurrentVeh_temp->StartTime)
                                                {
                                                    g_NewTripVehPointer->next= CurrentVeh_temp;
                                                    CurrentVeh_temp_front->next=g_NewTripVehPointer;
                                                    cont = FALSE;
                                                }
                                                else    if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                {
        if(g_NewTripVehPointer->orgZone<=CurrentVeh_temp->orgZone) //OrgZone?
                                                    {
        g_NewTripVehPointer->next= CurrentVeh_temp;
        CurrentVeh_temp_front->next=g_NewTripVehPointer;
                                                            cont = FALSE;
                                                    }
                                                  else
```

```
                                                                {
                                                                        cont = TRUE;
                                                                        while
((CurrentVeh_temp!=NULL) && cont)
                                                                        {
        if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                                                {
        if(g_NewTripVehPointer->orgZone<=CurrentVeh_temp->orgZone)
        {
        g_NewTripVehPointer->next=CurrentVeh_temp;
        CurrentVeh_temp_front->next=g_NewTripVehPointer;
        cont = FALSE;
                                                                                }
        else {
        CurrentVeh_temp_front=CurrentVeh_temp;
                CurrentVeh_temp=CurrentVeh_temp->next;
                                                                                }
                                                                        }
                                                                        else
//insert the dummy veh to the last postion with the same starttime
                                                                        {
        g_NewTripVehPointer->next=CurrentVeh_temp;
        CurrentVeh_temp_front->next=g_NewTripVehPointer;
        cont = FALSE;
                                                                        }
                                                                }       //While
((CurrentVeh_temp!=NULL) && cont)
                                                                }//else
                                                        }//else  if(g_NewTripVehPointer-
>StartTime==CurrentVeh_temp->StartTime)
                                                                } //while ((CurrentVeh_temp!=NULL) &&
cont)
                                                }       //else          if(g_NewTripVehPointer-
>StartTime>g_VehPointer ->StartTime)
                                                g_NewTripVehPointer=g_LastNewTripVehPtr-
>next;
                                                } //if(g_NewTripVehPointer!=NULL)
                                        } //if(! cont0)
                                }//end else
                        }//end if(destIndex==VehArrive->WorkLocation)
                        else if(destIndex==VehArrive->HomeLocation) //Dest = Home
```

```
		{
			VehArrive->fm_next->NumOfArrivedVehs+=1;
			if(VehArrive->fm_next->NumOfArrivedVehs == (VehArrive->fm_next->NumOfVehs-1))
			{
				//Generate a consolidated trip with departure time = evacuaiton time + delay, dest = nearest shelter
				if(g_NewTripVehPointer == NULL)
				{
					qps_GUI_printf("No new trip is ready to be generated for this veh!!!\n");
				}
				else
				{
					g_NewTripVehPointer->StartTime=current_time+VehArrive->DelayTime;
					g_NewTripVehPointer->desZone = pp_consolidated_dest(VehArrive->HomeLocation);
					g_NewTripVehPointer->orgZone = currentZonetemp;
					g_NewTripVehPointer->DelayTime = 0.0;
					g_NewTripVehPointer->ChangeTime = 0.0;
				  g_nEnRouteVehs--;
				  qps_DRW_vehicleColour(vehicle, API_ORANGE);

					//Insert this new vehicle into the sorted vehicle linked list

if(g_LastNewTripVehPtr!=NULL&&g_NewTripVehPointer!=NULL)
					{
						if((g_NewTripVehPointer->StartTime>g_LastNewTripVehPtr->StartTime))
							{
							  cont_trip=1;
							  if(g_NewTripVehPointer->next!=NULL&&g_NewTripVehPointer->next->StartTime==g_max_Time_3)
							  {

	g_LastNewTripVehPtr=g_NewTripVehPointer;

g_NewTripVehPointer=g_NewTripVehPointer->next;
							  }
							  else if (g_NewTripVehPointer->next==NULL)
							  {

	g_LastNewTripVehPtr=g_NewTripVehPointer;
								g_NewTripVehPointer=NULL;

							  }
							}
					}//end if(g_LastNewTripVehPtr!=NULL&&g_NewTripVehPointer!=NULL)
					if(! cont0)
					{
						//Step 1: isolate the new generated veh from the dummy vehs list

if(g_LastNewTripVehPtr!=NULL&&g_NewTripVehPointer!=NULL)
```

```
                                        {
                                          g_LastNewTripVehPtr-
>next=g_NewTripVehPointer->next;
                                        }
                                    //Step 2: Insert the isolated new generated veh to the
sorted veh link list
                                    if(g_NewTripVehPointer!=NULL)
                                        {
                                          if(g_NewTripVehPointer->StartTime
<g_VehPointer ->StartTime)
                                          { // insert front
                                                g_NewTripVehPointer->next=
g_VehPointer;
                                                   g_VehPointer=g_NewTripVehPointer;
                                          }
                                          else                    if(g_NewTripVehPointer-
>StartTime==g_VehPointer ->StartTime)
                                          {
                                                if(g_NewTripVehPointer-
>orgZone<=g_VehPointer->orgZone) //OrgZone?
                                                      {
                                                        g_NewTripVehPointer->next=
g_VehPointer;
g_VehPointer=g_NewTripVehPointer;
                                                      }
                                                   else
                                                      {
CurrentVeh_temp=g_VehPointer;
CurrentVeh_temp_front=CurrentVeh_temp;
                                                        cont = TRUE;
                                                        while
((CurrentVeh_temp!=NULL) && cont)
                                                        {
        if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                                {
if(g_NewTripVehPointer->orgZone<=CurrentVeh_temp->orgZone)
                                                                  {
        g_NewTripVehPointer->next=CurrentVeh_temp;
CurrentVeh_temp_front->next=g_NewTripVehPointer;
                                                                        cont
= FALSE;
                                                                  }
                                                                  else {
        CurrentVeh_temp_front=CurrentVeh_temp;
CurrentVeh_temp=CurrentVeh_temp->next;
                                                                  }
                                                                }
```

```
//if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                    else     //insert     the
dummy veh to the last postion with the same starttime
                                                        {
g_NewTripVehPointer->next=CurrentVeh_temp;
CurrentVeh_temp_front->next=g_NewTripVehPointer;
                                                          cont = FALSE;
                                                        }
                                                  }                    //While
((CurrentVeh_temp!=NULL) && cont)
                                                }       //if(g_NewTripVehPointer-
>orgZone<=g_VehPointer->orgZone)
                                        }//else          if(g_NewTripVehPointer-
>StartTime==g_VehPointer ->StartTime)
                                        else            if(g_NewTripVehPointer-
>StartTime>g_VehPointer ->StartTime)
                                        {
                CurrentVeh_temp=g_VehPointer;
CurrentVeh_temp_front=CurrentVeh_temp;
                                              cont = TRUE;
                                          while ((CurrentVeh_temp!=NULL) &&
cont)
                                              {
                                                if(g_NewTripVehPointer-
>StartTime>CurrentVeh_temp->StartTime)
                                                {
        CurrentVeh_temp_front=CurrentVeh_temp;
CurrentVeh_temp=CurrentVeh_temp->next;
                                                }
                                                else   if(g_NewTripVehPointer-
>StartTime<CurrentVeh_temp->StartTime)
                                                {
                                                        g_NewTripVehPointer-
>next= CurrentVeh_temp;
CurrentVeh_temp_front->next=g_NewTripVehPointer;
                                                          cont = FALSE;
                                                }
                                                else   if(g_NewTripVehPointer-
>StartTime==CurrentVeh_temp->StartTime)
                                                {
        if(g_NewTripVehPointer->orgZone<=CurrentVeh_temp->orgZone) //OrgZone?
                                                        {
g_NewTripVehPointer->next= CurrentVeh_temp;
CurrentVeh_temp_front->next=g_NewTripVehPointer;
                                                          cont = FALSE;
                                                        }
                                                  else
```

```
                                                                    {
                                                                      cont = TRUE;
                                                                      while
((CurrentVeh_temp!=NULL) && cont)
                                                                      {
        if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
        {
if(g_NewTripVehPointer->orgZone<=CurrentVeh_temp->orgZone)
{
        g_NewTripVehPointer->next=CurrentVeh_temp;
           CurrentVeh_temp_front->next=g_NewTripVehPointer;
           cont = FALSE;
         }
else {
        CurrentVeh_temp_front=CurrentVeh_temp;
                CurrentVeh_temp=CurrentVeh_temp->next;
         }
                                                                                }
                                                                          else
//insert the dummy veh to the last postion with the same starttime
        {
g_NewTripVehPointer->next=CurrentVeh_temp;
CurrentVeh_temp_front->next=g_NewTripVehPointer;
cont = FALSE;
                                                                                }
                                                                      }     //While
((CurrentVeh_temp!=NULL) && cont)
                                                                    }//else
                                                                }//else
if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                            }                   //while
((CurrentVeh_temp!=NULL) && cont)
                                                    }       //else      if(g_NewTripVehPointer-
>StartTime>g_VehPointer ->StartTime)
                                                      g_NewTripVehPointer=g_LastNewTripVehPtr-
>next;
                                                    } //if(g_NewTripVehPointer!=NULL)
```

```
                                } //if(! cont0)
                        }//end else
                }//end   if(VehArrive->fm_next->NumOfArrivedVehs   ==   (VehArrive->fm_next->NumOfVehs-1))
        }//end else if(destIndex==VehArrive->HomeLocation)
        else  //Dest = Random
        {
                //Generate a "meanderer" vehicle with random destination
                if(g_NewTripVehPointer == NULL)
                {
                        qps_GUI_printf("No new trip is ready to be generated for this veh!!!\n");
                }
                else
                {
                        do {
                                randomZone  =  LocationGenerator(17,18);    //Generate  work location for this vehicle
                        }while (randomZone == destIndex);
                        g_NewTripVehPointer->orgZone = currentZonetemp;
                   g_NewTripVehPointer->desZone = randomZone;
                   g_NewTripVehPointer->StartTime = current_time;
                        g_NewTripVehPointer->DelayTime = 0.0;
                        g_NewTripVehPointer->ChangeTime = 0.0;
                        g_NewTripVehPointer->fm_next->HomeBound  =  VehArrive->fm_next->HomeBound;
                        g_nEnRouteVehs--;
                        qps_DRW_vehicleColour(vehicle, API_ORANGE);


                        //Insert this new vehicle into the sorted vehicle linked list
                                if(g_LastNewTripVehPtr!=NULL&&g_NewTripVehPointer!=NULL)
                                {
                                        if((g_NewTripVehPointer->StartTime>g_LastNewTripVehPtr->StartTime))
                                        {
                                                cont_trip=1;
                                                if(g_NewTripVehPointer->next!=NULL&&g_NewTripVehPointer->next->StartTime==g_max_Time_3)
                                                {

        g_LastNewTripVehPtr=g_NewTripVehPointer;

        g_NewTripVehPointer=g_NewTripVehPointer->next;
                                                }
                                                else if (g_NewTripVehPointer->next==NULL)
                                                {

        g_LastNewTripVehPtr=g_NewTripVehPointer;
                                                        g_NewTripVehPointer=NULL;

                                                }
                                        }
                                }//end if(g_LastNewTripVehPtr!=NULL&&g_NewTripVehPointer!=NULL)
                                if(! cont0)
                                {
```

```
                                        //Step 1: isolate the new generated veh from the dummy
vehs list

if(g_LastNewTripVehPtr!=NULL&&g_NewTripVehPointer!=NULL)
                                        {
                                                g_LastNewTripVehPtr-
>next=g_NewTripVehPointer->next;
                                        }
                                        //Step 2: Insert the isolated new generated veh to the
sorted veh link list
                                        if(g_NewTripVehPointer!=NULL)
                                        {
                                                if(g_NewTripVehPointer->StartTime
<g_VehPointer ->StartTime)
                                                { // insert front
                                                        g_NewTripVehPointer->next=
g_VehPointer;
                                                        g_VehPointer=g_NewTripVehPointer;
                                                }
                                                else                    if(g_NewTripVehPointer-
>StartTime==g_VehPointer ->StartTime)
                                                {
                                                        if(g_NewTripVehPointer-
>orgZone<=g_VehPointer->orgZone) //OrgZone?
                                                        {
                                                                g_NewTripVehPointer->next=
g_VehPointer;

        g_VehPointer=g_NewTripVehPointer;
                                                        }
                                                        else
                                                        {

        CurrentVeh_temp=g_VehPointer;

        CurrentVeh_temp_front=CurrentVeh_temp;
                                                                cont = TRUE;
                                                                while
((CurrentVeh_temp!=NULL) && cont)
                                                                {

        if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                                        {

        if(g_NewTripVehPointer->orgZone<=CurrentVeh_temp->orgZone)
                                                                                {

        g_NewTripVehPointer->next=CurrentVeh_temp;

        CurrentVeh_temp_front->next=g_NewTripVehPointer;
                                                                                        cont    =
FALSE;
                                                                                }
                                                                                else {

        CurrentVeh_temp_front=CurrentVeh_temp;
```

```
        CurrentVeh_temp=CurrentVeh_temp->next;
                                                                    }
                                                                }
//if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                                else //insert the dummy
veh to the last postion with the same starttime
                                                                {
        g_NewTripVehPointer->next=CurrentVeh_temp;
        CurrentVeh_temp_front->next=g_NewTripVehPointer;
                                                                    cont = FALSE;
                                                                }
                                                            }                   //While
((CurrentVeh_temp!=NULL) && cont)
                                                        }               //if(g_NewTripVehPointer-
>orgZone<=g_VehPointer->orgZone)
                                                }//else                   if(g_NewTripVehPointer-
>StartTime==g_VehPointer ->StartTime)
                                                else                    if(g_NewTripVehPointer-
>StartTime>g_VehPointer ->StartTime)
                                                {
                                                    CurrentVeh_temp=g_VehPointer;
        CurrentVeh_temp_front=CurrentVeh_temp;
                                                    cont = TRUE;
                                                    while ((CurrentVeh_temp!=NULL) &&
cont)
                                                    {
                                                        if(g_NewTripVehPointer-
>StartTime>CurrentVeh_temp->StartTime)
                                                        {
        CurrentVeh_temp_front=CurrentVeh_temp;
        CurrentVeh_temp=CurrentVeh_temp->next;
                                                        }
                                                        else    if(g_NewTripVehPointer-
>StartTime<CurrentVeh_temp->StartTime)
                                                        {
                                                            g_NewTripVehPointer-
>next= CurrentVeh_temp;
                                                            CurrentVeh_temp_front-
>next=g_NewTripVehPointer;
                                                            cont = FALSE;
                                                        }
                                                        else    if(g_NewTripVehPointer-
>StartTime==CurrentVeh_temp->StartTime)
                                                        {
        if(g_NewTripVehPointer->orgZone<=CurrentVeh_temp->orgZone) //OrgZone?
                                                            {
        g_NewTripVehPointer->next= CurrentVeh_temp;
```

```
        CurrentVeh_temp_front->next=g_NewTripVehPointer;
                                                                cont = FALSE;
                                                        }
                                                else
                                                        {

                                                                cont = TRUE;
                                                                while
((CurrentVeh_temp!=NULL) && cont)
                                                                {

        if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                                        {

        if(g_NewTripVehPointer->orgZone<=CurrentVeh_temp->orgZone)

        {

        g_NewTripVehPointer->next=CurrentVeh_temp;

        CurrentVeh_temp_front->next=g_NewTripVehPointer;

        cont = FALSE;
                                                                                }

        else {

        CurrentVeh_temp_front=CurrentVeh_temp;

                CurrentVeh_temp=CurrentVeh_temp->next;
                                                                                }
                                                                        }
                                                                        else
//insert the dummy veh to the last postion with the same starttime
                                                                        {

        g_NewTripVehPointer->next=CurrentVeh_temp;

        CurrentVeh_temp_front->next=g_NewTripVehPointer;

        cont = FALSE;
                                                                        }
                                                                }       //While
((CurrentVeh_temp!=NULL) && cont)
                                                        }//else
                                                }//else  if(g_NewTripVehPointer-
>StartTime==CurrentVeh_temp->StartTime)
                                        } //while ((CurrentVeh_temp!=NULL) &&
cont)
                                }       //else          if(g_NewTripVehPointer-
>StartTime>g_VehPointer ->StartTime)

                                g_NewTripVehPointer=g_LastNewTripVehPtr-
>next;

                        } //if(g_NewTripVehPointer!=NULL)
```

```
                                } //if(! cont0)
                        }//end else
                  }//end else
            }//end if(current_time<g_Evacuation_Start_Time-0.05)
            else //Evacuation started!
            {
                  if(VehArrive->ChangeTime >= current_time){
                        //this vehicle's change time >= current time
                        if(VehArrive->NewDest == g_dummy_shelter) {
                              //Generate a new trip with origin = current location, dest =
nearest shelter, release time = current time
                              if(g_NewTripVehPointer == NULL)
                              {
                                    qps_GUI_printf("No new trip is ready to be generated for
this veh!!!\n");
                              }
                              else
                              {
                                    g_NewTripVehPointer->orgZone = currentZonetemp;
                                    g_NewTripVehPointer->desZone                  =
pp_consolidated_dest(currentZonetemp);
                                    g_NewTripVehPointer->StartTime = current_time;
                                    g_NewTripVehPointer->DelayTime = 0.0;
                                    g_NewTripVehPointer->ChangeTime = 0.0;
                                    g_NewTripVehPointer->AwareTime                =
g_NewTripVehPointer->StartTime;
                                    g_NewTripVehPointer->fm_next->HomeBound       =
VehArrive->fm_next->HomeBound;

                                    g_nEnRouteVehs--;
                                    qps_DRW_vehicleColour(vehicle, API_ORANGE);

                                    //Insert this new vehicle into the sorted vehicle linked list

            if(g_LastNewTripVehPtr!=NULL&&g_NewTripVehPointer!=NULL)
                                    {
                                          if((g_NewTripVehPointer-
>StartTime>g_LastNewTripVehPtr->StartTime))
                                          {
                                                cont_trip=1;
                                                if(g_NewTripVehPointer-
>next!=NULL&&g_NewTripVehPointer->next->StartTime==g_max_Time_3)
                                                {

            g_LastNewTripVehPtr=g_NewTripVehPointer;

            g_NewTripVehPointer=g_NewTripVehPointer->next;
                                                }
                                                else      if      (g_NewTripVehPointer-
>next==NULL)
                                                {

            g_LastNewTripVehPtr=g_NewTripVehPointer;
                                                      g_NewTripVehPointer=NULL;

                                                }
```

```
                                        }
                                }//end
if(g_LastNewTripVehPtr!=NULL&&g_NewTripVehPointer!=NULL)
                                if(! cont0)
                                {
                                        //Step 1: isolate the new generated veh from the
dummy vehs list

if(g_LastNewTripVehPtr!=NULL&&g_NewTripVehPointer!=NULL)
                                        {
                                                g_LastNewTripVehPtr-
>next=g_NewTripVehPointer->next;
                                        }
                                        //Step 2: Insert the isolated new generated veh
to the sorted veh link list
                                         if(g_NewTripVehPointer!=NULL)
                                        {
                                                if(g_NewTripVehPointer->StartTime
<g_VehPointer ->StartTime)
                                                { // insert front
                                                        g_NewTripVehPointer->next=
g_VehPointer;

        g_VehPointer=g_NewTripVehPointer;
                                                }
                                                else            if(g_NewTripVehPointer-
>StartTime==g_VehPointer ->StartTime)
                                                {
                                                        if(g_NewTripVehPointer-
>orgZone<=g_VehPointer->orgZone) //OrgZone?
                                                        {
                                                                g_NewTripVehPointer-
>next= g_VehPointer;

        g_VehPointer=g_NewTripVehPointer;
                                                        }
                                                        else
                                                        {


        CurrentVeh_temp=g_VehPointer;

        CurrentVeh_temp_front=CurrentVeh_temp;
                                                                cont = TRUE;
                                                                while
((CurrentVeh_temp!=NULL) && cont)
                                                                {

        if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                                        {

        if(g_NewTripVehPointer->orgZone<=CurrentVeh_temp->orgZone)
                                                                                {

        g_NewTripVehPointer->next=CurrentVeh_temp;

        CurrentVeh_temp_front->next=g_NewTripVehPointer;
```

```
        cont = FALSE;
                                                                    }
                                                                    else {

        CurrentVeh_temp_front=CurrentVeh_temp;

        CurrentVeh_temp=CurrentVeh_temp->next;
                                                                    }
                                                                }
//if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                                else //insert the
dummy veh to the last postion with the same starttime
                                                                {

        g_NewTripVehPointer->next=CurrentVeh_temp;

        CurrentVeh_temp_front->next=g_NewTripVehPointer;
                                                                    cont =
FALSE;
                                                                }
                                                            } //While
((CurrentVeh_temp!=NULL) && cont)
                                                        } //if(g_NewTripVehPointer-
>orgZone<=g_VehPointer->orgZone)
                                                    }//else if(g_NewTripVehPointer-
>StartTime==g_VehPointer ->StartTime)
                                                    else if(g_NewTripVehPointer-
>StartTime>g_VehPointer ->StartTime)
                                                    {

        CurrentVeh_temp=g_VehPointer;

        CurrentVeh_temp_front=CurrentVeh_temp;
                                                        cont = TRUE;
                                                        while
((CurrentVeh_temp!=NULL) && cont)
                                                        {

        if(g_NewTripVehPointer->StartTime>CurrentVeh_temp->StartTime)
                                                            {

        CurrentVeh_temp_front=CurrentVeh_temp;

        CurrentVeh_temp=CurrentVeh_temp->next;
                                                            }
                                                            else
if(g_NewTripVehPointer->StartTime<CurrentVeh_temp->StartTime)
                                                            {

        g_NewTripVehPointer->next= CurrentVeh_temp;

        CurrentVeh_temp_front->next=g_NewTripVehPointer;
                                                                cont = FALSE;
                                                            }
                                                            else
```

```
if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                                    {
        if(g_NewTripVehPointer->orgZone<=CurrentVeh_temp->orgZone) //OrgZone?
                                                                        {
        g_NewTripVehPointer->next= CurrentVeh_temp;
        CurrentVeh_temp_front->next=g_NewTripVehPointer;
                                                                        cont    =
FALSE;
                                                                        }
                                                                    else
                                                                        {
                                                                        cont    =
TRUE;
                                                                        while
((CurrentVeh_temp!=NULL) &&  cont)
                                                                        {
        if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
        {
        if(g_NewTripVehPointer->orgZone<=CurrentVeh_temp->orgZone)
        {
                g_NewTripVehPointer->next=CurrentVeh_temp;
                CurrentVeh_temp_front->next=g_NewTripVehPointer;
                cont = FALSE;
        }
        else {
                CurrentVeh_temp_front=CurrentVeh_temp;
                CurrentVeh_temp=CurrentVeh_temp->next;
        }
                                                                        }
        else //insert the dummy veh to the last postion with the same starttime
        {
        g_NewTripVehPointer->next=CurrentVeh_temp;
        CurrentVeh_temp_front->next=g_NewTripVehPointer;
        cont = FALSE;
                                                                        }
```

```
                                                                                }
//While ((CurrentVeh_temp!=NULL) && cont)
                                                                        }//else
                                                                }//else
if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                                }                       //while
((CurrentVeh_temp!=NULL) && cont)
                                                        }       //else      if(g_NewTripVehPointer-
>StartTime>g_VehPointer ->StartTime)


        g_NewTripVehPointer=g_LastNewTripVehPtr->next;

                                                        } //if(g_NewTripVehPointer!=NULL)
                                                } //if(! cont0)
                                        }//end else{
                                }//if(VehArrive->NewDest == g_dummy_shelter)
                                else
                                {
                                        //Generate a new trip in which origin = current, dest = newdest,
startTime = current time,
                                        if(currentZonetemp == VehArrive->HomeLocation)
                                        {
                                                VehArrive->fm_next->NumOfArrivedVehs++;
                                                g_nEnRouteVehs--;
                                                VehArrive->ChangeTime = 0.0;
                                                VehArrive->AwareTime = current_time;
                                        }
                                        else
                                        {
                                                if(g_NewTripVehPointer == NULL)
                                                {
                                                        qps_GUI_printf("No new trip is ready to be
generated for this veh!!!\n");
                                                }
                                                else
                                                {
                                                        g_NewTripVehPointer->orgZone                =
currentZonetemp;
                                                        g_NewTripVehPointer->desZone                =
g_NewTripVehPointer->NewDest;//new destnation is home
                                                       g_NewTripVehPointer->StartTime = current_time;
                                                        g_NewTripVehPointer->AwareTime              =
g_NewTripVehPointer->StartTime;
                                                        g_NewTripVehPointer->fm_next->HomeBound =
VehArrive->fm_next->HomeBound;
                                                        g_nEnRouteVehs--;
                                                        qps_DRW_vehicleColour(vehicle,
API_ORANGE);

                                                        //Insert this new vehicle into the sorted vehicle
linked list

        if(g_LastNewTripVehPtr!=NULL&&g_NewTripVehPointer!=NULL)
                                                        {
                                                                if((g_NewTripVehPointer-
```

```
>StartTime>g_LastNewTripVehPtr->StartTime))
                                                {
                                                        cont_trip=1;
                                                        if(g_NewTripVehPointer-
>next!=NULL&&g_NewTripVehPointer->next->StartTime==g_max_Time_3)
                                                        {

        g_LastNewTripVehPtr=g_NewTripVehPointer;

        g_NewTripVehPointer=g_NewTripVehPointer->next;
                                                        }
                                                        else if (g_NewTripVehPointer-
>next==NULL)
                                                        {

        g_LastNewTripVehPtr=g_NewTripVehPointer;

        g_NewTripVehPointer=NULL;
                                                        }
                                                }//end           if((g_NewTripVehPointer-
>StartTime...
                                        }//end
if(g_LastNewTripVehPtr!=NULL&&g_NewTripVehPointer!=NULL)

                                        if(! cont0)
                                        {
                                                //Step 1: isolate the new generated veh
from the dummy vehs list

if(g_LastNewTripVehPtr!=NULL&&g_NewTripVehPointer!=NULL)
                                                {
                                                        g_LastNewTripVehPtr-
>next=g_NewTripVehPointer->next;
                                                }
                                                //Step 2: Insert the isolated new
generated veh to the sorted veh link list
                                                 if(g_NewTripVehPointer!=NULL)
                                                {
                                                        if(g_NewTripVehPointer-
>StartTime <g_VehPointer ->StartTime)
                                                        { // insert front
                                                                g_NewTripVehPointer-
>next= g_VehPointer;

        g_VehPointer=g_NewTripVehPointer;
                                                        }
                                                        else     if(g_NewTripVehPointer-
>StartTime==g_VehPointer ->StartTime)
                                                        {

        if(g_NewTripVehPointer->orgZone<=g_VehPointer->orgZone) //OrgZone?
                                                                {

        g_NewTripVehPointer->next= g_VehPointer;

        g_VehPointer=g_NewTripVehPointer;
```

```
                                                                    }
                                                                    else
                                                                    {
        CurrentVeh_temp=g_VehPointer;
        CurrentVeh_temp_front=CurrentVeh_temp;
                                                                        cont = TRUE;
                                                                        while
((CurrentVeh_temp!=NULL) && cont)
                                                                        {
        if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                                            {
        if(g_NewTripVehPointer->orgZone<=CurrentVeh_temp->orgZone)
        {
        g_NewTripVehPointer->next=CurrentVeh_temp;
        CurrentVeh_temp_front->next=g_NewTripVehPointer;
        cont = FALSE;
                                                                                }
        else {
        CurrentVeh_temp_front=CurrentVeh_temp;
        CurrentVeh_temp=CurrentVeh_temp->next;
                                                                                }
                                                                            }
//if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                                            else
//insert the dummy veh to the last postion with the same starttime
                                                                            {
        g_NewTripVehPointer->next=CurrentVeh_temp;
        CurrentVeh_temp_front->next=g_NewTripVehPointer;
        cont = FALSE;
                                                                            }
                                                                        } //While
((CurrentVeh_temp!=NULL) && cont)
                                                                    }
//if(g_NewTripVehPointer->orgZone<=g_VehPointer->orgZone)
                                                        }//else if(g_NewTripVehPointer-
>StartTime==g_VehPointer ->StartTime)
                                                        else if(g_NewTripVehPointer-
>StartTime>g_VehPointer ->StartTime)
                                                        {
        CurrentVeh_temp=g_VehPointer;
```

```
        CurrentVeh_temp_front=CurrentVeh_temp;
                                                                cont = TRUE;
                                                                while
((CurrentVeh_temp!=NULL) && cont)
                                                                {
        if(g_NewTripVehPointer->StartTime>CurrentVeh_temp->StartTime)
                                                                        {
        CurrentVeh_temp_front=CurrentVeh_temp;
        CurrentVeh_temp=CurrentVeh_temp->next;
                                                                        }
                                                                        else
if(g_NewTripVehPointer->StartTime<CurrentVeh_temp->StartTime)
                                                                        {
        g_NewTripVehPointer->next= CurrentVeh_temp;
        CurrentVeh_temp_front->next=g_NewTripVehPointer;
                                                                                cont    =
FALSE;
                                                                        }
                                                                        else
if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                                        {
        if(g_NewTripVehPointer->orgZone<=CurrentVeh_temp->orgZone) //OrgZone?
                                                                                {
        g_NewTripVehPointer->next= CurrentVeh_temp;
        CurrentVeh_temp_front->next=g_NewTripVehPointer;
        cont = FALSE;
                                                                                }
                                                                           else
                                                                                {


        cont = TRUE;
        while ((CurrentVeh_temp!=NULL) && cont)
        {
        if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
        {
                if(g_NewTripVehPointer->orgZone<=CurrentVeh_temp->orgZone)
                {
                        g_NewTripVehPointer->next=CurrentVeh_temp;
```

```
                        CurrentVeh_temp_front->next=g_NewTripVehPointer;
                        cont = FALSE;
                }
                else {
                        CurrentVeh_temp_front=CurrentVeh_temp;
                        CurrentVeh_temp=CurrentVeh_temp->next;
                }
        }
        else //insert the dummy veh to the last postion with the same starttime
        {
                g_NewTripVehPointer->next=CurrentVeh_temp;
                CurrentVeh_temp_front->next=g_NewTripVehPointer;
                cont = FALSE;
        }
                                                                                }
//While ((CurrentVeh_temp!=NULL) && cont)
                                                                        }//else
                                                                }//else
if(g_NewTripVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                                }       //while
((CurrentVeh_temp!=NULL) && cont)
                                                        } //else if(g_NewTripVehPointer-
>StartTime>g_VehPointer ->StartTime)

        g_NewTripVehPointer=g_LastNewTripVehPtr->next;
                                                                } //if(g_NewTripVehPointer!=NULL)
                                                        } //if(! cont0)

                                                                }//end else
                                                        }//end else
                                                }//end else
                        }//end if(VehArrive->ChangeTime >= current_time)
                        else //this vehicle's change time < current time, which means this vehicle has
already been changed its destination while it is en route
                        {
                                if(destIndex==VehArrive->HomeLocation) //if the vehicle's dest equals to
its home
                                {
                                        if(VehArrive->fm_next->HomeBound==1) // this vehicle is a
consolidated Veh
```

```
{
	//The following codes handles consolidated vehicles' consolidation process

	FamArrive=VehArrive->fm_next;  //point to its associated family

	//CheckNotes: 4. How many vehs in this family, increament arrival veh by 1
	num_vhc_arrive=FamArrive->NumOfArrivedVehs+1;
//	qps_GUI_printf("At time %5.2f, a vehicle from Org %d arrived at Dest %d, to make %d out of %d family vehicles\n", qpg_CFG_simulationTime(), qpg_VHC_origin(vehicle), qpg_VHC_destination(vehicle), num_vhc_arrive, FamArrive->NumOfVehs);

	FamArrive->NumOfArrivedVehs=num_vhc_arrive;

	//CheckNotes: 5. If # arrived vehs = # vehs, dispath consolidated trip
	if(FamArrive->NumOfArrivedVehs == (FamArrive->NumOfVehs-1)) // if total number of arrived veh equal to the num of veh of the family)
	{
	  //Debugging
	//if(qpg_VHC_destination(vehicle) == 97 && qpg_VHC_origin(vehicle)==408) //&& qpg_VHC_startTime(vehicle)==2445.5)
	//	iidebug=1;

	//Determine the start time of the dummy veh (consolidated veh) according to awareness time of each vehicle in this family
	  minTime=g_max_Time;
	for (icounter; icounter<=FamArrive->NumOfArrivedVehs; icounter++)
	{
		if(FamArrive->VehicleList!=NULL) {
			if(FamArrive->VehicleList->AwareTime<minTime)
				minTime=FamArrive->VehicleList->AwareTime;

				FamArrive->VehicleList=FamArrive->VehicleList->next;
		}
	}
	if(minTime<=current_time)
		maxTime=current_time;
	else
	   maxTime = minTime;

	//Assign the start time to the dummy veh generated at the PP_Init
	arrival_time = maxTime;


	// Generate Delay Time for each vehicle, randomly distributed at [0, 30min]
	 r = pp_Uniform(delay_start_time, delay_end_time);
```

```
                                        if(g_DummyVehPointer==NULL)
                                                qps_GUI_printf("no Family ready to be
assigned a dummy veh!!!\n");
                                        else
                                        {
                                                g_DummyVehPointer->DelayTime = r;
                                                g_DummyVehPointer->StartTime = arrival_time
+ r;

                                                //Determine which destination the "dummy
(consolidated)" vehicle will be assigned
                                                ConsolidatedDestination =
pp_consolidated_dest(g_DummyVehPointer->HomeLocation);
                                                g_DummyVehPointer->desZone
=ConsolidatedDestination;  //the new destination zone , temp
                                                //qps_GUI_printf("All veh arrived, a dummy
veh %d is ready to release to the network, start time %5.2f, home %d, orig %d, dest %d\n",
        g_DummyVehPointer->VehicleID, g_DummyVehPointer->StartTime,
g_DummyVehPointer->HomeLocation, g_DummyVehPointer->orgZone, g_DummyVehPointer-
>desZone);


                                                //Generate outputs into a txt file
                                                g_Results=fopen(fname,"a");
                                                fprintf(g_Results,"All veh arrived, a dummy
veh %d is ready to release to the network, start time %5.2f, home %d, orig %d, dest %d\n",
        g_DummyVehPointer->VehicleID, g_DummyVehPointer->StartTime,
g_DummyVehPointer->HomeLocation, g_DummyVehPointer->orgZone, g_DummyVehPointer-
>desZone);
                                                fclose(g_Results);

                                                // Insert the new generated veh to the sorted veh
link list.
                                                //Step 0: Check the new generated veh from the
dummy vehs list is larger than the last regular veh

        if(g_LastDummyVehPointer!=NULL&&g_DummyVehPointer!=NULL)
                                                {
                                                        if((g_DummyVehPointer-
>StartTime>g_LastDummyVehPointer->StartTime))
                                                        {
                                                                cont0=1;
                                                                if(g_DummyVehPointer-
>next!=NULL&&g_DummyVehPointer->next->StartTime==g_max_Time)
                                                                {

        g_LastDummyVehPointer=g_DummyVehPointer;

        g_DummyVehPointer=g_DummyVehPointer->next;
                                                        //              cont0=1;
                                                                }
                                                                else if (g_DummyVehPointer-
>next==NULL)
                                                                {

        g_LastDummyVehPointer=g_DummyVehPointer;
```

```
	g_DummyVehPointer=NULL;
                                                        }
                                                    }
                                                }
//if(g_LastDummyVehPointer!=NULL&&g_DummyVehPointer!=NULL)

                                                if(! cont0)
                                                {
                                                    //Step 1: isolate the new generated veh
from the dummy vehs list

if(g_LastDummyVehPointer!=NULL&&g_DummyVehPointer!=NULL)
                                                    {
                                                        g_LastDummyVehPointer-
>next=g_DummyVehPointer->next;
                                                    }
                                                    //Step 2: Insert the isolated new
generated veh to the sorted veh link list
                                                    if(g_DummyVehPointer!=NULL)
                                                    {
                                                        if(g_DummyVehPointer-
>StartTime <g_VehPointer ->StartTime)
                                                        { // insert front
                                                            g_DummyVehPointer-
>next= g_VehPointer;

	g_VehPointer=g_DummyVehPointer;
                                                        }
                                                        else     if(g_DummyVehPointer-
>StartTime==g_VehPointer ->StartTime)
                                                        {
                                                            if(g_DummyVehPointer-
>orgZone<=g_VehPointer->orgZone) //OrgZone?
                                                            {

	g_DummyVehPointer->next= g_VehPointer;

	g_VehPointer=g_DummyVehPointer;
                                                            }
                                                            else
                                                            {


	CurrentVeh_temp=g_VehPointer;

	CurrentVeh_temp_front=CurrentVeh_temp;
                                                                cont = TRUE;
                                                                while
((CurrentVeh_temp!=NULL) && cont)
                                                                {

	if(g_DummyVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                                    {

	if(g_DummyVehPointer->orgZone<=CurrentVeh_temp->orgZone)
```

```
        {
        g_DummyVehPointer->next=CurrentVeh_temp;
        CurrentVeh_temp_front->next=g_DummyVehPointer;
        cont = FALSE;
                                                                    }
        else {
        CurrentVeh_temp_front=CurrentVeh_temp;
        CurrentVeh_temp=CurrentVeh_temp->next;
                                                                    }
                                                                }
//if(g_DummyVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                                else
//insert the dummy veh to the last postion with the same starttime
                                                                {
        g_DummyVehPointer->next=CurrentVeh_temp;
        CurrentVeh_temp_front->next=g_DummyVehPointer;
        cont = FALSE;
                                                                }
                                                        } //While
((CurrentVeh_temp!=NULL) && cont)
                                                    }
//if(g_DummyVehPointer->orgZone<=g_VehPointer->orgZone)
                                                }//else if(g_DummyVehPointer-
>StartTime==g_VehPointer ->StartTime)
                                                else if(g_DummyVehPointer-
>StartTime>g_VehPointer ->StartTime)
                                                {
        CurrentVeh_temp=g_VehPointer;
        CurrentVeh_temp_front=CurrentVeh_temp;
                                                    cont = TRUE;
                                                    while
((CurrentVeh_temp!=NULL) && cont)
                                                    {
        if(g_DummyVehPointer->StartTime>CurrentVeh_temp->StartTime)
                                                        {
        CurrentVeh_temp_front=CurrentVeh_temp;
        CurrentVeh_temp=CurrentVeh_temp->next;
                                                        }
                                                        else
if(g_DummyVehPointer->StartTime<CurrentVeh_temp->StartTime)
                                                        {
```

```
        g_DummyVehPointer->next= CurrentVeh_temp;
        CurrentVeh_temp_front->next=g_DummyVehPointer;
                                                                    cont    =
FALSE;
                                                                }
                                                                else
if(g_DummyVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                                {
        if(g_DummyVehPointer->orgZone<=CurrentVeh_temp->orgZone) //OrgZone?
                                                                    {
        g_DummyVehPointer->next= CurrentVeh_temp;
        CurrentVeh_temp_front->next=g_DummyVehPointer;
        cont = FALSE;
                                                                    }
                                                                else
                                                                    {

        cont = TRUE;
        while ((CurrentVeh_temp!=NULL) && cont)
        {
        if(g_DummyVehPointer->StartTime==CurrentVeh_temp->StartTime)
        {
                if(g_DummyVehPointer->orgZone<=CurrentVeh_temp->orgZone)
                {
                        g_DummyVehPointer->next=CurrentVeh_temp;
                        CurrentVeh_temp_front->next=g_DummyVehPointer;
                        cont = FALSE;
                }
                else {
                        CurrentVeh_temp_front=CurrentVeh_temp;
                        CurrentVeh_temp=CurrentVeh_temp->next;
                }
        }
        else //insert the dummy veh to the last postion with the same starttime
```

```
        {
                g_DummyVehPointer->next=CurrentVeh_temp;
                CurrentVeh_temp_front->next=g_DummyVehPointer;
                cont = FALSE;
        }
                                                                                                }
//While ((CurrentVeh_temp!=NULL) && cont)
                                                                                        }//else
                                                                                }//else
if(g_DummyVehPointer->StartTime==CurrentVeh_temp->StartTime)
                                                                        }               //while
((CurrentVeh_temp!=NULL) && cont)
                                                                    } //else if(g_DummyVehPointer-
>StartTime>g_VehPointer ->StartTime)

        g_DummyVehPointer=g_LastDummyVehPointer->next;
                                                        } //if(g_DummyVehPointer!=NULL)
                                                } //if(! cont0)
                                        }//else {
                                }       //if(FamArrive->NumOfArrivedVehs      ==      (FamArrive-
>NumOfVehs-1))
                        } //if(VehArrive->fm_next->HomeBound==1)

                        else //this vehicle is a non-consolidated Veh
                        {
                                //set the vehicle's destination to the closest shelter
                                VehArrive->desZone      =       pp_consolidated_dest(VehArrive-
>HomeLocation);
                                //set the vehicle's departure time to the current time
                                VehArrive->StartTime = current_time;
                        }
                } //if(destIndex==VehArrive->HomeLocation)
                else
                {
                        //Vehicle arrive at shelter, collect statistical data, end of story.
                        iidebug=1;


                        g_Results=fopen(fname1,"a");
                        fprintf(g_Results,"At time %5.2f a vehicle with homebound %d fam
members %d from Org %d arrived at Dest %d \n", qpg_CFG_simulationTime(), VehArrive-
>fm_next->HomeBound,        VehArrive->fm_next->NumOfVehs,        qpg_VHC_origin(vehicle),
qpg_VHC_destination(vehicle));
                        fclose(g_Results);

                }
          } //if (VehArrive->ChangeTime >= current_time)
```

```
	}//	if(current_time<g_Evacuation_Start_Time)


	//Debug
	//Count the total number of vehicles arrived at shelters

	if(destIndex==613 || destIndex ==614 || destIndex ==615 || destIndex ==616)
	{
		g_ArrivedShelterVehicle++;
	}


}



/* ---------------------------------------------------------------------
 * This function sets the Network Structure, the Family Linked List and the Vehicle Linked List Structures
 * --------------------------------------------------------------------- */
static void pp_init_data(void)
{

	//NET_USERDATA *netdata = NULL;
	struct Family *CurrentFamily;
	struct Vehicle *CurrentVehicle;
	// Veh Point always point to the end of the veh list of each family
	struct Vehicle *ptr_Vehicle_tail=NULL;
	int i, j, dummyFamilyIndex=0, ThisHomeLoc, ThisWorkLoc, index=0;
	int veh_id=0, i_index, fam_id=0;
	struct Vehicle *tempVehiclePtr;
	int TotalVehicles = 0;
	float* Veh_Start_Time;
	int vehTotal=0, temp_HomeLocation=0;
	float vehStart_time=0, r;
	float delay_start_time = 0;
	float delay_end_time = 1800; //300 seconds = 5 mins
	int temp_dummy_veh=0, tripType=0, randomZone=0, randomZone2=0;
	VEHICLE* vehicle=NULL;

	netdata = calloc(1, sizeof(NET_USERDATA));
	netdata->NumOfFamilies = g_nFamilies;

	CurrentFamily = calloc(1,sizeof(struct Family));
	netdata->FamilyList = CurrentFamily;

// Build Family and veh link list
	for (i=1; i<=netdata->NumOfFamilies; i++)
	{
		CurrentFamily->NumOfArrivedVehs = 0;

		//Generate home locaton for the current family (home location ~ normal distribution)
		ThisHomeLoc = LocationGenerator(17, 18);  //the links in the network are 17 * 18

		CurrentFamily->HomeLocation = ThisHomeLoc;
```

```
		//Generate number of vehicles for the current family
	CurrentFamily->NumOfVehs = VehicleGenerator();

		//Generate homebound attribute for the current family
		CurrentFamily->HomeBound = HomeboundGenerator();

		//Genearte family ID for the current family
		fam_id++;
		CurrentFamily->FamilyID = fam_id;

		if(CurrentFamily->HomeBound == 1) {
		  CurrentFamily->NumOfVehs = CurrentFamily->NumOfVehs+1; // +1 a dummy veh without departure time
		}

		TotalVehicles = TotalVehicles + CurrentFamily->NumOfVehs;

		//Allocate memory for vehicle list
		CurrentVehicle = calloc(1,sizeof(struct Vehicle));
		if(ptr_Vehicle_tail!=NULL)
			ptr_Vehicle_tail->next=CurrentVehicle;
	//	CurrentVehicle->next=NULL;
		CurrentFamily->VehicleList=CurrentVehicle;

		//generate vehicle list (its capacity and its origin) for the current family
		for (j=1; j<=CurrentFamily->NumOfVehs; j++){
			CurrentVehicle->capacity = VehicleCapacityGenerator();

// Check the generated work location (Origin Zone) is the same as the home location (Dest zone)
// if Origin Zone == Destination Zone, generate a new orgin zone, else assign other properties to the veh
			do {
				ThisWorkLoc = LocationGenerator(17,18);	//Generate work location for this vehicle
			} while (ThisWorkLoc==ThisHomeLoc);

			CurrentVehicle->WorkLocation = ThisWorkLoc;
			CurrentVehicle->HomeLocation = CurrentFamily->HomeLocation;

			//Generate Vehicle ID for each vehicle
			veh_id++;
			CurrentVehicle->VehicleID = veh_id;

			//Set org and dest zone for the current veh
			tripType = VehicleODGenerator();
			if (tripType == 1) { //Home-Work Trip
				CurrentVehicle->orgZone=CurrentVehicle->HomeLocation;
				CurrentVehicle->desZone=CurrentVehicle->WorkLocation;
			}
			else if (tripType == 2) { //Work-Home Trip
				CurrentVehicle->orgZone = CurrentVehicle->WorkLocation;
				CurrentVehicle->desZone = CurrentVehicle->HomeLocation;
			}
			else if (tripType == 3) { //stay at home trip
				CurrentVehicle->orgZone = CurrentVehicle->HomeLocation;
```

```
//origin is home
                                CurrentVehicle->desZone = g_dummy_home;  //destination is a
dummy destination (zone 750)
                                //CurrentVehicle->vehType = 1;
                        }
                        else if (tripType == 4) //Home-Random Trip
                        {
                                CurrentVehicle->orgZone=CurrentVehicle->HomeLocation;
                                do{
                                        randomZone = LocationGenerator(17,18);   //Generate
random destination for this vehicle
                                }while (randomZone == CurrentVehicle->orgZone || randomZone
== CurrentVehicle->WorkLocation);
                                CurrentVehicle->desZone = randomZone;
                        }
                        else //Random-Random trip
                        {
                                do {
                                        randomZone = LocationGenerator(17,18);
                                }while (randomZone == CurrentVehicle->orgZone || randomZone
== CurrentVehicle->WorkLocation);
                                CurrentVehicle->orgZone = randomZone;
                                do {
                                        randomZone2 = LocationGenerator(17,18);
                                }while    (randomZone2    ==    CurrentVehicle->orgZone    ||
randomZone2 == CurrentVehicle->WorkLocation || randomZone2 == randomZone);
                                CurrentVehicle->desZone = randomZone2;
                        }


                        //Current Veh Point back to the family
                        CurrentVehicle->fm_next=CurrentFamily;

                        if ( j < CurrentFamily->NumOfVehs) {
                                CurrentVehicle->next = calloc(1,sizeof(struct Vehicle));
                                //Current Veh Point back to the family
                                CurrentVehicle->fm_next=CurrentFamily;
                                CurrentVehicle = CurrentVehicle->next;
                                //CurrentVehicle->next=NULL;
                        }
                        if(j==CurrentFamily->NumOfVehs)
                                CurrentVehicle->next=NULL;
                           ptr_Vehicle_tail=CurrentVehicle;
                }

                if (i<g_nFamilies)
                {
                        CurrentFamily->next = calloc(1,sizeof(struct Family));
                        CurrentFamily = CurrentFamily->next;
                }

                else if(i==g_nFamilies) //family link list tail point to NULL, add by Ke at 6/22/09
                {
                        CurrentFamily->NumOfVehs = g_nNewTripDummyVehs;
                        TotalVehicles = TotalVehicles + CurrentFamily->NumOfVehs;
```

```
                        //Allocate memory for vehicle list
                        CurrentVehicle = calloc(1,sizeof(struct Vehicle));
                        if(ptr_Vehicle_tail!=NULL)
                                ptr_Vehicle_tail->next=CurrentVehicle;
                        CurrentFamily->VehicleList=CurrentVehicle;

                        //generate a dummy vehicle list for this dummy family
                        for (j=1; j<=CurrentFamily->NumOfVehs; j++){
                                CurrentVehicle->orgZone = CurrentFamily->HomeLocation;
                                CurrentVehicle->desZone = g_dummyDest;
                                veh_id++;
                                CurrentVehicle->VehicleID = veh_id;
                                CurrentVehicle->fm_next=CurrentFamily;

                                if ( j < CurrentFamily->NumOfVehs) {
                                        CurrentVehicle->next = calloc(1,sizeof(struct Vehicle));
                                        CurrentVehicle->fm_next=CurrentFamily;
                                        CurrentVehicle = CurrentVehicle->next;
                                }

                                if(j==CurrentFamily->NumOfVehs) {
                                        CurrentVehicle->next=NULL;
                                        ptr_Vehicle_tail=CurrentVehicle;
                                }
                        } //end for (j=1; j<=CurrentFamily->NumOfVehs; j++)
                        CurrentFamily->next=NULL;
                   CurrentVehicle->next=NULL; // Veh Link List tail point to NULL
                }//end else if(i==g_nFamilies)
        } //end for (i=1; i<=netdata->NumOfFamilies; i++)


   //Assign the start time and delay time to each vehicle
   Veh_Start_Time=(float*) calloc(TotalVehicles, sizeof(float));
        pp_start_time_assign(TotalVehicles,Veh_Start_Time);
   CurrentFamily=netdata->FamilyList; //Starting from the head of the family link list
        for (i=1; i<g_nFamilies; i++)
        {
                CurrentVehicle=CurrentFamily->VehicleList;
                for (j=1; j<=CurrentFamily->NumOfVehs; j++){  //Scan each veh of every family
                        CurrentVehicle->StartTime = Veh_Start_Time[index];
                        // Generate Delay Time for each vehicle, randomly distributed at [0,
15min]
                        r = pp_Uniform(delay_start_time, delay_end_time);
                        CurrentVehicle->DelayTime=r;

                        //Assign the dummy veh only to the family having more than 1 veh
                        if(CurrentFamily-
>NumOfVehs>1&&CurrentFamily!=NULL&&j==CurrentFamily->NumOfVehs) {
                                CurrentVehicle->StartTime=g_max_Time;
                        }

                        //Assign a large start time to stay-at-home vehicle
                        if(CurrentVehicle->desZone == g_dummy_home
&&CurrentFamily!=NULL) {
                                CurrentVehicle->StartTime=g_max_Time_2;
                        }
```

```
                        if(CurrentVehicle->StartTime==g_max_Time)
                                temp_dummy_veh++;

                        CurrentVehicle = CurrentVehicle->next;
                        index++;
                } //end for (j=1; j<=CurrentFamily->NumOfVehs; j++)

                CurrentFamily=CurrentFamily->next;

        }//end for (i=1; i<=g_nFamilies; i++)


        //Generate a large departure time for dummy vehicles used for new trip generation, add by Ke on 6/22/09
        if(i==g_nFamilies){
          CurrentVehicle=CurrentFamily->VehicleList;
          for (j=1; j<=CurrentFamily->NumOfVehs; j++){
            CurrentVehicle->StartTime = g_max_Time_3;
                  CurrentVehicle = CurrentVehicle->next;
                  temp_dummy_veh++;
          }

        }


/* This is used to generate initlization report. It should be comment out when necessary
        sprintf(fname,"c:/After_Init.dat");
        g_Results_2=fopen(fname,"w");
        tempVehiclePtr=netdata->FamilyList->VehicleList;
        while (tempVehiclePtr != NULL)
        {
                //qps_GUI_printf("\nVehicle %d, origin %d, destination %d (%d vehicles in this family), release time %5.2f\n", VehID, tempVehiclePtr->WorkLocation, tempFamilyPtr->HomeLocation, tempFamilyPtr->NumOfVehs, tempVehiclePtr->StartTime );
        fprintf(g_Results_2,"Vehicle %d release time %7.2f origin %d destination %d in family %d\n", tempVehiclePtr->VehicleID, tempVehiclePtr->StartTime, tempVehiclePtr->orgZone, tempVehiclePtr->desZone, tempVehiclePtr->fm_next->FamilyID);
                tempVehiclePtr = tempVehiclePtr->next;
        }
        fclose(g_Results_2);
End Comment */

/*debug
        sprintf(fname,"c:/Init_without_dummy.dat");
        g_Results=fopen(fname,"w");
        tempVehiclePtr=netdata->FamilyList->VehicleList;
        while (tempVehiclePtr != NULL)
        {
                if (tempVehiclePtr->StartTime != g_max_Time) {
                        //qps_GUI_printf("\nVehicle %d, origin %d, destination %d (%d vehicles in this family), release time %5.2f\n", tempVehiclePtr->VehicleID, tempVehiclePtr->WorkLocation, tempFamilyPtr->HomeLocation, tempFamilyPtr->NumOfVehs, tempVehiclePtr->StartTime );
                fprintf(g_Results,"Vehicle %d release time %7.2f origin %d destination %d delay time %7.2f\n", veh_id, tempVehiclePtr->StartTime, tempVehiclePtr->orgZone, tempVehiclePtr->desZone, tempVehiclePtr->DelayTime);
```

```
                    }
                    tempVehiclePtr = tempVehiclePtr->next;
          }
          fclose(g_Results);
debug*/

          qps_GUI_printf("vehicle list has been built\n");

/* VehType Initalization for visaulazation check */
   for (i=0;i<40;i++)
             scenarioType[i]=FALSE;


}

//Bubble Sort for the veh_linked_list
static void pp_sort_data(void)
{

          struct Family *CurrentFamily;
          struct Vehicle *CurrentVehicle;
   struct Vehicle *lst, *tmp , *prev, *potentialprev, *head ;
          int idx, idx2, TotalVeh = 0;
          int  vehTotal=0,vehStart_time=0, dummy_veh=0, new_veh=0;
          BOOL next_dummyVeh = FALSE;
          BOOL next_newVeh = FALSE;
          int veh_id=0, i_index;
          struct Vehicle *tempVehiclePtr, *tempVehiclePtr_1;


          //Traverse the veh list and change the start time
   CurrentVehicle=netdata->FamilyList->VehicleList;
          while (CurrentVehicle !=NULL)
          {
       vehTotal++;
              vehStart_time=CurrentVehicle->StartTime;
                    if(vehStart_time==g_max_Time)
                              dummy_veh++;
                    CurrentVehicle=CurrentVehicle->next;
          }

          CurrentVehicle=netdata->FamilyList->VehicleList;
          head=CurrentVehicle;
          potentialprev=CurrentVehicle;

  //determine total number of nodes
  for (tmp=CurrentVehicle;tmp; tmp=tmp->next)
  {
                    TotalVeh++;
                    vehStart_time=tmp->StartTime;
  }

 for (idx=0; idx<TotalVeh-1; idx++)
 {
          for (idx2=0,lst=head;lst && lst->next && (idx2<=TotalVeh-1-idx); idx2++)
   {
```

```
                    if (!idx2)
                    {
      //we are at beginning, so treat start
      //node as prev node
                              prev = lst;
                    }

    //compare the two neighbors
    if (lst->next->StartTime  < lst->StartTime)
    {
      //swap the nodes
      tmp = (lst->next?lst->next->next:0);

      if (!idx2 && (prev == head))
      {
        //we do not have any special sentinal nodes
        //so change beginning of the list to point
        //to the smallest swapped node
        head = lst->next;
      }
      potentialprev = lst->next;
      prev->next = lst->next;
      lst->next->next = lst;
      lst->next = tmp;
      prev = potentialprev;
    }
    else
    {
      lst = lst->next;
      if(idx2)
      {
        //just keep track of previous node,
        //for swapping nodes this is required
        prev = prev->next;
      }
    } //else
  } //for (idx2=0,lst=head;lst && lst->next && (idx2<=TotalVeh-1-idx); idx2++)
} // for (idx=0; idx<TotalVeh-1; idx++)
 netdata->FamilyList->VehicleList=head;

//double-sort by zone
          CurrentVehicle=netdata->FamilyList->VehicleList;
          head=CurrentVehicle;
          potentialprev=CurrentVehicle;

  for (idx=0; idx<TotalVeh-1; idx++)
          {
                    for (idx2=0,lst=head;lst && lst->next && (idx2<=TotalVeh-1-idx); idx2++)
                    {
                              if (!idx2)
                              {
                                        //we are at beginning, so treat start
                                        //node as prev node
                                        prev = lst;
                              }
                              //compare the two neighbors
```

```
                        if ((lst->next->orgZone < lst->orgZone) && (lst->next->StartTime == lst->StartTime))
                        {
                                //swap the nodes
                                tmp = (lst->next?lst->next->next:0);

                                if (!idx2 && (prev == head))
                                {
                                //we do not have any special sentinal nodes
                                //so change beginning of the list to point
                                //to the smallest swapped node
                                        head = lst->next;
                                }
                                potentialprev = lst->next;
                                prev->next = lst->next;
                                lst->next->next = lst;
                                lst->next = tmp;
                                prev = potentialprev;
                        }
                        else
                        {
                                lst = lst->next;
                                if(idx2)
                                {
                                //just keep track of previous node,
                                //for swapping nodes this is required
                                        prev = prev->next;
                                }
                        }
                }
        }
   netdata->FamilyList->VehicleList=head;
//end double sort-Jan 5 2009


//Traverse the veh list
   CurrentVehicle=netdata->FamilyList->VehicleList;
        g_LastDummyVehPointer=CurrentVehicle;
        g_LastNewTripVehPtr=CurrentVehicle;

        vehTotal=0;
        dummy_veh=0;
        while (CurrentVehicle !=NULL)
        {
      vehTotal++;
            vehStart_time=CurrentVehicle->StartTime;

                 if(g_LastDummyVehPointer->next->StartTime==g_max_Time
&&dummy_veh==0) {
                //Reach the end of the veh link list (before the dummy veh)
                        next_dummyVeh=TRUE;
                 }
                 if(!next_dummyVeh)
                        g_LastDummyVehPointer=CurrentVehicle;

                 if(vehStart_time==g_max_Time)
```

```
                    dummy_veh++;
                if(dummy_veh == 1)
                    g_DummyVehPointer=CurrentVehicle;     //g_DummyVehPointer point to
the head of the veh link list

                //Add by Ke on 6/23/09
                if(g_LastNewTripVehPtr->next->StartTime == g_max_Time_3 && new_veh==0) {
                    next_newVeh = TRUE;
                }
                if(!next_newVeh)
                    g_LastNewTripVehPtr=CurrentVehicle;
                if(vehStart_time == g_max_Time_3)
                    new_veh++;
                if(new_veh == 1)
                    g_NewTripVehPointer=CurrentVehicle;

                CurrentVehicle=CurrentVehicle->next;
        }

// This is used to generate a report after doubl-sorting to check double-sorting's result. It should
be comment out when necessary
        sprintf(fname,"c:/After_D-Sort.dat");
        g_Results_2=fopen(fname,"w");
        tempVehiclePtr=netdata->FamilyList->VehicleList;
        while (tempVehiclePtr != NULL)
        {
                //qps_GUI_printf("\nVehicle %d, origin %d, destination %d (%d vehicles in this
family), release time %5.2f\n", VehID, tempVehiclePtr->WorkLocation, tempFamilyPtr-
>HomeLocation, tempFamilyPtr->NumOfVehs, tempVehiclePtr->StartTime );
        fprintf(g_Results_2,"Vehicle %d release time %7.2f, origin %d, dest %d, home %d,
work %d, familyID %d, homebound %d, delay %7.2f, vehicle type %d, change time %7.2f, new
dest %d, and num of arrived veh %d\n", tempVehiclePtr->VehicleID, tempVehiclePtr->StartTime,
tempVehiclePtr->orgZone, tempVehiclePtr->desZone, tempVehiclePtr->HomeLocation,
tempVehiclePtr->WorkLocation, tempVehiclePtr->fm_next->FamilyID, tempVehiclePtr->fm_next-
>HomeBound, tempVehiclePtr->DelayTime, tempVehiclePtr->vehType, tempVehiclePtr-
>ChangeTime, tempVehiclePtr->NewDest, tempVehiclePtr->fm_next->NumOfArrivedVehs);

                tempVehiclePtr = tempVehiclePtr->next;
        }
        fclose(g_Results_2);
//End Comment*/

        qps_GUI_printf("vehicle list has been sorted\n");
}

//This function determines which destination zone is for each consolidated vehicle
static int pp_consolidated_dest (int curr_home )
{
        int x,y,cons_dest;
   int z1,z2;


//traffic flows go to four shelters
        if(curr_home<=306) {
                y = curr_home % 17;
           x = ((curr_home - y) / 17) ;
```

```
                if(y>=x) {
                        if((17-y)>=x)
                                cons_dest = g_boundaryZone_West;
                        else
                                cons_dest = g_boundaryZone_South;
                        }
                else {
                        if((18-x)>=y)
                                cons_dest = g_boundaryZone_North;
                        else
                                cons_dest = g_boundaryZone_East;
                        }
                }
        else {
                y = curr_home % 18;
           x = ((curr_home - y-306) / 18) +1;
                if(y>=x) {
                        if((18-y)>=x)
                                cons_dest = g_boundaryZone_West;
                        else
                                cons_dest = g_boundaryZone_South;
                        }
                else {
                        if((17-x)>=y)
                                cons_dest = g_boundaryZone_North;
                        else
                                cons_dest = g_boundaryZone_East;

                }
        }

        return cons_dest;


}

//print-out the Family & Vehicle list
static void pp_print_data(void)
{

        struct Family *tempFamilyPtr;
        struct Vehicle *tempVehiclePtr;
        int VehID = 0;

        sprintf(fname, "c:/InitFile.dat");
        g_Results_2=fopen(fname, "w");
        tempVehiclePtr=netdata->FamilyList->VehicleList;
        while (tempVehiclePtr != NULL)
        {
                VehID++;
                tempFamilyPtr = tempVehiclePtr->fm_next;
                //qps_GUI_printf("\nVehicle %d, origin %d, destination %d (%d vehicles in this family), release time %5.2f\n", VehID, tempVehiclePtr->WorkLocation, tempFamilyPtr->HomeLocation, tempFamilyPtr->NumOfVehs, tempVehiclePtr->StartTime );
        fprintf(g_Results_2,"Vehicle %d has home %d, origin %d, destination %d, release time %5.2f\n", tempVehiclePtr->VehicleID, tempVehiclePtr->HomeLocation, tempVehiclePtr-
```

```
>orgZone, tempVehiclePtr->desZone, tempVehiclePtr->StartTime);

                tempVehiclePtr = tempVehiclePtr->next;
        }
        fclose(g_Results_2);
}

/*Generate home/work locations from normal random variates*/
long LocationGenerator(int rowNum, int columnNum)
{
        double probNum;
        float x, y;
        int m, n, w, thisLocation;

        //Generate a pair of normal random variates, which locate on [-3,3]
        do{
                pp_NormalRandomNum(&x,&y);
        }while ((x<-3||x>3)||(y<-3||y>3));


       // x and y are ~normal(0,1).  They need to be scaled, and discretized into link ID numbers.
        // m and y are scaled on [-8,8]

   //Generate a Bernoulli number
        probNum = ((double)rand()/((double)(RAND_MAX)+(double)(1))); /* random value in
range [0,1) */
   w = pp_Bernoulli(probNum);

        //If w=o, this location is located on one of those E-W zones;
        //if w=1, this location is located on one of those N-S zones;
        if (w == 0) {
                m = (int)((columnNum / 6) * (x + 3));
                n = (int)((y + 3) * rowNum/6);
                thisLocation = m * rowNum + n + 1;
        }
        else {
                m = (int)((x + 3) * rowNum/6);
                n = (int)((columnNum / 6) * (y + 3));
                thisLocation = rowNum * columnNum + m * columnNum + n + 1;
        }
        return thisLocation;
}

/*For each family i, generate the random number of vehicles (0, 1, 2, 3).*/
/*Assume that 50% families have 1 vehicle, 40% of them have 2, and 10% of them have 3.*/
int VehicleGenerator(void)
{
        double p;
        int NumOfGeneratedVehs;

        p = ((double)rand()/((double)(RAND_MAX)+(double)(1))); /* random value in range [0,1)
*/

        if (p<0.5) {
                NumOfGeneratedVehs = 1;
        }
```

```
	else if (p<0.9){
		NumOfGeneratedVehs = 2;
	}
	else
		NumOfGeneratedVehs = 3;


	return NumOfGeneratedVehs;
}

//Generate Veh Type 0--- Individual, 1--- consolidated veh
int VehTypeGenerator(void)
{
	double p;
	int TypeOfGeneratedVehs;

	p = ((double)rand()/((double)(RAND_MAX)+(double)(1))); /* random value in range [0,1)
*/
	if (p < 0.5) {
		TypeOfGeneratedVehs = 0;  //Individual veh
	}
	else if (p<1) {
		TypeOfGeneratedVehs = 1; //Consolidated Veh
	}

	return TypeOfGeneratedVehs;
}

/*For each vehicle i, generate its origin and destination*/
/*Assume that: 50% vehicles have home-work trips, 40% vehilces have work-home trips, and
10% vehicles have home-random trips.*/
int VehicleODGenerator(void)
{
	double p;
	int tripTypes;

	p = ((double)rand()/((double)(RAND_MAX)+(double)(1))); /* random value in range [0,1)
*/

	if (p<0.4) {
		tripTypes = 1; //Home-Work trip
	}
	else if (p<0.8){
		tripTypes = 2; //Work-Home Trip
	}
	else if (p<0.85){
		tripTypes = 3;  //stay-at-home trip
	}
	else if (p<0.9){
		tripTypes = 4; //Home-Random Trip
	}
	else
		tripTypes = 5; //Random-Random trip

	return tripTypes;
```

```
}

/*For each vehicle i, generate its capacity, (2,4,6,8) person/each*/
/*Assume that: 15% vehicles have 2-person capacity, 50% - 4 persons, 25% - 6 persons, and 10% - 8 persons.*/
int VehicleCapacityGenerator(void)
{
	double p;
	int vehCapacity;

	p = ((double)rand()/((double)(RAND_MAX)+(double)(1))); /* random value in range [0,1) */
	if (p < 0.15) {
		vehCapacity = 2;
	}
	else if (p<0.65) {
		vehCapacity = 4;
	}
	else if (p<0.9){
		vehCapacity = 6;
	}
	else
		vehCapacity = 8;

	return vehCapacity;
}

/*For each family, generate its homebound attribute*/
/*Assume that: 50% families have home-bound trip (family-dependent), while the other 50% families are not family-dependent.*/
int HomeboundGenerator(void)
{
	double x;
	int HomeboundFamily;

	x = ((double)rand()/((double)(RAND_MAX)+(double)(1))); /* random value in range [0,1) */
	if (x < 0.5) {
		HomeboundFamily = 0;  //non-homebound family

	}
	else if (x<1) {
		HomeboundFamily = 1; //homebound family
	}

	return HomeboundFamily;
}

//Random Number Generators

/*Generate N(0,1) random variates using Polar Method*/
void pp_NormalRandomNum(float *x1, float *x2)
{
	float y, w, v1, v2;
	double r1, r2;
```

```
        do
   {
      r1= ((double)rand()/((double)(RAND_MAX)+(double)(1)));   /* uniform random value in range
[0,1) */
      r2= ((double)rand()/((double)(RAND_MAX)+(double)(1)));   /* uniform random value in range
[0,1) */
      v1  = 2.0 * r1 - 1.0;
      v2  = 2.0 * r2 - 1.0;
      w = v1 * v1 + v2 * v2;
         }while (w>=1.0);

   y = sqrt (-2.0 * log (w) / w);

   *x1 = v1 * y;
   *x2 = v2 * y;
}

/*Generate a Bernoulli random number, which returns 1 with prob p or 0 with prob 1-p.*/
long pp_Bernoulli(double p)
{
        double r;

   r= ((double)rand()/((double)(RAND_MAX)+(double)(1)));   /* random value in range [0,1) */

   if (r < (1.0 - p))
                return 0;
        else
                return 1;
}

/*Generate a Triangular random number */
double pp_triangular(double min, double max, double mode)
{
        double unifNum;
        double x;

        unifNum = ((double)rand()/(RAND_MAX+1));      //Generate a uniform number in [0,1]

        if(unifNum <= ((mode-min)/(max-min))) {
                x = min + sqrt(unifNum*(max-min)*(mode-min));
        }
        else
        {
                x = max - sqrt((1-unifNum)*(max-min)*(max-mode));
        }

        return x;
}

/*Generate a random number in a specific range. */
float pp_Uniform (float lowest, float highest)
{
        double r;
        float r_range, range;
   r= ((double)rand()/((double)(RAND_MAX)+(double)(1)));   /* r is a random number in range [0,1)
*/
```

```
	range = (highest - lowest) + 1;
	r_range = lowest + (range*r);
	return r_range;
}


//Veh Start Time Generation: 0-3hrs
void pp_start_time_assign(int Total_Veh, float* Veh_Start_Time)
{
	int i, Veh_1;
	double Time_min, Time_max, Time_mode;
	Time_min = 0.0;
	Time_max = 18000.0; //5-hr simulation period
	Time_mode = 10800.0;
	Veh_1 = Total_Veh;
	for (i = 0; i < Veh_1; i++){
		Veh_Start_Time[i] = pp_triangular(Time_min,Time_max,Time_mode);
	}
}

/* ---------------------------------------------------------------------
 * This function sets all the variables needed for releasing a vehicle into
 * our network.
 * --------------------------------------------------------------------- */
static void pp_release_vehicle(int dest)
{

   /* normal will be used as the distrution for our DVU's aggression and
    * awarness factors */
   int normal[9] =  {1, 4, 11, 21, 26, 21, 11, 4, 1};
   int aggr;
   int awar;
   int sum;
   int new_sum;
   /* maximum integer size */
   int max_rand = 32767;
   int i;

	//Add by Ke, we can change vehicle type later if required.
	int type = 1;

   /* this callback function set the type of vehicle to be released from the
    * zone */
   qps_ZNE_vehicleType(type);
   /* this callback function set the destination zone index of the vehicle
    * about to be released */
   qps_ZNE_vehicleDestination(dest);


   /* calculate aggression and awareness factors */
   aggr = (((float)rand()/(float)(max_rand)) *100.0);
   awar = (((float)rand()/(float)(max_rand)) *100.0);

   sum = 0;
   for(i = 0; i < 9; i++)
   {
```

```
        new_sum = sum + normal[i];

        if((aggr > sum) && (aggr <= new_sum))
        {
           qps_ZNE_vehicleAggression(i);
        }

        if((awar > sum) && (awar <= new_sum))
        {
           qps_ZNE_vehicleAwareness(i);
        }

        sum = new_sum;
     }
}

void pp_draw_yet_to_be_released_vehicles(void)
{
          struct Vehicle *tempVehiclePtr;
          Bool cont=TRUE;

          tempVehiclePtr = g_VehList_headPointer;
          while (tempVehiclePtr != NULL)
          {
                    //qps_DRW_forceVisableObjectsRebuild(TRUE);
                    if(tempVehiclePtr->orgZone == tempVehiclePtr->WorkLocation &&
tempVehiclePtr->desZone == tempVehiclePtr->HomeLocation && tempVehiclePtr->fm_next-
>HomeBound == 1 && tempVehiclePtr->StartTime>g_Evacuation_Start_Time+tempVehiclePtr-
>DelayTime && cont){
                              //qps_DRW_vehicleTag(veh_temp, API_RED, 2,3, "scenario 13..");
                              cont = FALSE;
                    }

                    tempVehiclePtr = tempVehiclePtr->next;
          }
}



void qpx_NET_complete(void)
{
          struct Vehicle *tempVehiclePtr;
          int VehID = 0;

          pp_print_data();

}
```

References

Alsnih, R., J. Rose, and P. Stopher (2005). “Understanding household evacuation decisions using a stated choice survey – case study of bush fires.” Proceedings of the 84th Annual Meeting of the Transportation Research Board, Washington D. C.

Balakrishna, R., Y. Wen, M. Ben-Akiva, and C. Antoniou (2008). “Simulation-based framework for transportation network management for emergencies.” Proceedings of the 87th Annual Meeting of the Transportation Research Board, Washington D. C.

Barrett, B., B. Ran, and R. Pillai (2000), “Developing a dynamic traffic management modeling framework for hurricane evacuation.” Transportation Research Record 1733, pp. 115-121.

Chien, S. I., and Korikanthimath, V. V. (2007). “Analysis and Modeling of Simultaneous and Staged Emergency Evacuations.” Journal of Transportation Engineering, Vol. 133, Issue 3, pp. 190-197.

Chiu, Y. C. (2004), “Traffic scheduling simulation and assignment for area-wide evacuation.” Proceedings on the 7th International IEEE Conference on ITS, pp. 537-542.

Chiu, Y. C., P. Korada, and P. B. Mirchandani (2005). “Dynamic traffic management for evacuation.” Proceedings of the 84th Annual Meeting of the Transportation Research Board, Washington D. C.

Chen, X. and F. B. Zhan (2008) “Agent-based modeling and simulation of urban evacuation: relative effectiveness of simultaneous and staged evacuation strategies”. Journal of the Operational Research Society 59, pp. 25-33.

Chen, M., Chen, L. and Miller-Hooks, E. (2007). “Traffic Signal Timing for Urban Evacuation.” Journal of Urban Planning and Development, Vol. 133, No. 1, pp.30-42.

Church, R. L. and Sexton, R. (2002). “Modeling small area evacuation: Can existing transportation infrastructure impede public safety?” Final representation, California Dept. of Transportation, Testbed Center for Interoperability, Sacramento, Calif.

Cova, Thomas J. and J.P. Johnson (2002), “Microsimulation of neighborhood evacuations in the urban – wildland interface.” Environment and Planning A 2002, Vol. 34, pp. 2211-2229.

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

Deitel, H. M. and P. J. Deitel (2001), “C++ How To Program, third edition”, Prentice Hall Publications.

Dow, K. and S. L. Cutter (2002), “Emerging hurricane evacuation issues: Hurricane Floyd and South Carolina”, Natural Hazards Review, Vol.3, pp.12-18.

Drabek, Thomas E. (1986), “Human System Responses to Disaster: An Inventory of Sociological Findings.”, Spring-Verlag, New York, NY.

Dunn, C. E., and Newton, D. (1992), “Optimal routes in GIS and emergency planning applications”, Area 24, 259-267

El-Mitiny N, S. Ramasamy and E. Radwan (2007). “Transit facilities’ emergency evacuation planning and preparedness using traffic simulation.” Proceedings of the 86th Annual Meeting of the Transportation Research Board, Washington D. C.

Farrell, J. (2005) “Alternatives to road building to improve hurricane evacuation in Coastal South Carolina.” Environmental Advocacy Seminar, available online at http://law.sc.edu/environmental/papers/200511/eas/farrell.pdf

Han, L. D. and Yuan, F. (2005). “Evacuation modeling and operations using dynamic traffic assignment and most desirable destination approaches.” Proceedings of the 84th Annual Meeting of the Transportation Research Board, Washington D. C.

Han, L.D., Y. Fang and U. II Thomas (2007). “What is an effective evacuation operation.” Journal of Urban Planning and Development, Vol. 133, Issue 1, pp. 3-8.

Hobeika, A. G., and Jamei, B. (1985). “MASSVAC: A model for calculating evacuation times under natural disaster.” Emergency planning, Simulation Series 15, pp. 23-28.

Hobeika, Antonie G. and Changkyun Kim (1998), “Comparison of traffic assignments in evacuation modeling.”, IEEE Transactions on Engineering Management, Vol.45, No.2, pp. 192-198.

ITT Industries (2000). “TSIS User’s Guide”, Version 5.0, prepared for Federal Highway Administration, January 2000.

Jha, M., K. Moore and B. Pashaie (2004). “Emergency evacuation planning with microscopic traffic simulation.” Proceedings of the 83rd Annual Meeting of the Transportation Research Board, Washington D. C.

KLD Associates. (1984). “Formulations of the DYNEV and I-DYNEV traffic simulation models used in ESF.” Federal Emergency Management Agency, Washington, D.C.

Goldblatt, R. (2004). “Evacuation planning: a key part of emergency planning.” Proceedings of the 83rd Annual Meeting of the Transportation Research Board, Washington D.C.

Kwon, E., and Pitt, S. (2005). “Evaluation of emergency evacuation strategies for downtown event traffic using a dynamic network model.” Transportation Research

Record. 1922, Transportation Research Board, pp. 149-155.

Laefer, D. F., A. R. Pradhan and A. Koss (2006). “GIS-based disaster management systems: a cogent data framework.” Proceedings of the 85th Annual Meeting of the Transportation Research Board, Washington D.C.

Lei, Y. C. and S. Y. Zhang (2004). “Features and partial derivatives of Bertalnaffy-Richards growth model in forestry.” Nonlinear analysis: modeling and control, Vol. 9, No. 1, pp 65 to 73.

Lim, E. and B. Wolshon (2005). “Modeling and performance assessment of contraflow evacuation termination points.” Transportation Research Record. 1922, Transportation Research Board, pp. 118-128.

Liu, H., J. X. Ban, W. Ma and P. B. Mirchandani (2007). “Model reference adaptive control framework for real-time traffic management under emergency evacuation” Journal of Urban Planning and Development, Vol. 133, Issue 1, pp. 43-50.

Liu, Y, N. Zou and G.L Chang (2005), “An integrated emergency evacuation system for real-time operations – A case study of Ocean City, Maryland under hurricane attacks.”, Proceedings of the 8th International IEEE Conference on Intelligent Transportation Systems, Vienna, Austria, September 13-16.

Liu, Y., X. Lai, and G-L Chang (2006). “A cell-based network optimization model for staged evacuation planning under emergencies.” Proceedings of the 85th Annual Meeting of the Transportation Research Board, Washington D. C.

Liu, Ying (2007). “An integrated optimal control system for emergency evacuation.” Ph.D. Thesis, University of Maryland, College Park, 2007.

Lively, D., Elhamshary, O., and Tournay W. (2006). “Advanced traveler information

system: an overview of California's system application and its performance as part of emergency response planning." Proceedings of the 85th Annual Meeting of the Transportation Research Board, Washington D. C.

Metropolitan Washington Council of Governments (MWCOG) (2004). "Regional emergency evacuation transportation coordination annex, regional emergency coordination plan." Available online at: http://www.mwcog.org/uploads/committee-documents/q15fXF820040315131526.pdf

Mitchell, S.W. and E. Radwan. (2006). "Heuristic prioritization of emergency evacuation staging to reduce clearance time." Proceedings of the 85th Annual Meeting of the Transportation Research Board, Washington D.C.

Morrow, R. B. (2002). "Implementing ITS for hurricane evacuations in Florida". ITE Journal, Vol. 72, Issue 4, pp. 46-50.

Murray-Tuite, P.M. and H.S. Mahmassani (2003), "Model of Household Trip-Chain Sequencing in Emergency Evacuation.", Transportation Research Record, Issue 1831, pp. 21-29.

Murray-Tuite, P. M. (2003). "Identification of vulnerable transportation infrastructure and household decision making under emergency evacuation conditions." Ph.D. Thesis, University of Texas, Austin, 2003.

Murray-Tuite, P.M. and H.S. Mahmassani (2004), "Transportation network evacuation planning with household activity interactions," Transportation Research Record, Issue 1894, pp.150-159.

Pal, A., A. J. Graettinger and M. H. Triche (2003). "Emergency evacuation modeling based on geographical information system data." Proceedings of the 82nd Annual Meeting of the Transportation Research Board, Washington D.C.

Pidd, M., F.N. de Silva, and R.W. Eglese (1996), “Theory and methodology, a simulation model for emergency evacuation.”, European Journal of Operation Research, Vol.90, Issue.3, pp.413-419.

Prater, C., D. Wenger, and K. Grady (2000), “Hurricane Bret post storm assessment: A review of the utilization of hurricane evacuation studies and information dissemination.”, Texas A&M University Hard Reduction and Recovery Center, College Station, TX.

PTV Associates, Inc. (2005). “VISSIM 4.1 Manual”, version 4.1, March 2005.

Quadstone Limited (2006), “Quadstone Paramics V5.2 Modeller Users Guide”, version No. 1.0, Public Distribution.

Quadstone Limited (2006), “Quadstone Paramics V5.2 Programmer Users Guide”, version No. 2.0, Public Distribution.

Quadstone Limited (2006), “Quadstone Paramics V5.2 Programmer Reference Manual”, Public Distribution.

Rathi, A.K., and Solanki, R.S. (1993). “Simulation of traffic flow during emergency evacuations: a microcomputer-based modeling system.” Proceedings of the 1993 Winter Simulation Conference, Los Angeles, U.S.

Richards, F. J. (1959). “A flexible growth function for empirical use.” Journal of Express Botanical, Vol. 10, pp 290 – 300.

RITA (Research and Innovative Technology Administration) (2001), “Highlights of the 2001 national household travel survey.” Available online at: http://www.bts.gov/publications/highlights_of_the_2001_national_household_travel_survey/html/executive_summary.html

Rontiris, K. and Crous, W. (2000). “Emergency evacuation modeling for the Koeberg Nuclear Power Station.” Proceeding 2nd Asian EMME12 User’s Meeting, Cape Metropolitan Council, Cape Town, South Afterica.

Sbayti, H. and H. Mahmassani (2006). “Optimal scheduling of evacuation operations.” TRB 85th Annual Meeting Compendium of Papers (CD-ROM), Transportation Research Board, Washington, D.C.

Sattayhatewa, P. and B. Ran. (2000), “Developing a dynamic traffic management model for nuclear power plant evacuation.”, presented at the 79th TRB meeting, Washington D.C.

Sheffi, Y., Mahmassani, H., and W. B. Powell (1982), “A Transportation Network Evacuation Model.”, Transportation Research – A., Vol. 16A, No. 3, pp 209-218.

Shashi, S. and S. Kim (2006). “Contraflow transportation network reconfiguration for evacuation route planning.” Minnesota Department of Transportation, Technical Report, MN/RC-2006-21.

Sinuany-Stern Z, Stern E. (1993), “Simulating the evacuation of a small city: the effects of traffic factors.”, Socio-Economic Planning Sciences 27 pp 97-108.

Silva, F.N., and R.W. Eglese (2000), “Integrating simulation modeling and GIS: spatial decision support systems for evacuation planning.”, Journal of the Operation Research Society, Vol. 51, No. 4, pp. 423-430.

Sisiopiku, V. P., Jones, S., Sullivan, A., Sullivan, A. J., Pathakar, S., and Tang, X. (2004). “Regional traffic simulation for emergency preparedness.” Technical Report 03326, University Transportation Center for Alabama.

Southworth, Frank (1991). “Regional evacuation modeling: a state-of-the-art review.”,

Oak Ridge National Laboratory, ORNL/TM-11740.

Tagliaferri, A. P. (2005). “Use and comparison of traffic simulation models in the analysis of emergency evacuation conditions.” M.S. Thesis, North Carolina University, 2005.

Theodoulou, G. and B. Wolshon (2004). “Modeling and analyses of freeway contraflow to improve future evacuations.” TRB 83rd Annual Meeting Compendium of Papers (CD-ROM), Transportation Research Board, Washington, D.C.

Tierney, K.J., M.K. Lindell, and R.W. Perry (2001), “Facing the unexpected: Diaster preparedness and response in the United States.”, Joseph Henry Press, Washington, D.C.

Tsoularis, A. and J. Wallace (2002), “Analysis of logistic growth models”, Mathematical biosciences, Vol. 179, pp. 21-55.

Tuydes, H. and Ziliaskopoulos, A. (2004). “Network re-design to optimize evacuation contraflow.” TRB 83rd Annual Meeting Compendium of Papers (CD-ROM), Transportation Research Board, Washington, D.C.

Tuydes, H. and Ziliaskopoulos, A. (2006). “A Tabu-based heuristic approach for the optimization of network evacuation contraflow.” TRB 85th Annual Meeting Compendium of Papers (CD-ROM), Transportation Research Board, Washington, D.C.

Urbanik, T.II. (1978). “Texas hurricane evacuation study.”, Texas Transportation Institute, College Station, Texas.

Urbanik, T. II., P.L. Cummings, and Desrosler, A.E. (1981). “An analysis of evacuation time estimates around 52 nuclear power plant sites.”, U.S. Nuclear Regulatorry Commission, NUREG/CR-1856, Volumes 1 and 2. Washington, D. C.

Urbanik, T.II. (2000). “Evacuation time estimates for nuclear power plants.”, Journal of

Hazardous Materials, Vol.75, pp.165-180.

U.S. Army Corps of Engineers (USACE). (1979). “Lee County, Florida flood emergency evacuation plan.” Dept. of the Army, and Southwest Florida Regional Planning Council.

Williams, B. M., Tagliaferri, A. P., Meinhold, S. S., Hummer, J. E., and Rouphail, N. M.(2007). “Simulation and analysis of freeway lane reversal for coastal hurricane evacuation.” Journal of Urban Planning and Development, Vol. 133, Issue 1, pp 61-72.

Wolshon, B. (2001). “One-way-out: Contraflow freeway operation for hurricane evacuation.”, Natural Hazards Review, Vol. 2, pp.105-112.

Wolshon, B. (2002). “Planning for the evacuation of New Orleans.” ITE Journal, Vol. 72, Issue 2, pp. 44-49.

Wolshon, B and L. Lambert. (2005). “Design of reversible roadway entry and termination points: comparative review of the state of practice.” TRB 84th Annual Meeting Compendium of Papers (CD-ROM), Transportation Research Board, Washington, D.C.

Zeigler, D. J., S.D. Brunn, and J.H. Johnson, Jr. (1981). “Evacuation from a nuclear technological disaster.”, Geographical Review, Vol.71, pp.1-16.

Zou, Nan, S.T. Yeh and G. L. Chang (2005), “Simulation-based emergency evacuation system for Ocean City Maryland during Hurricanes.” Transportation Research Record. 1922, Transportation Research Board, pp. 138-148.