ABSTRACT

Title of Dissertation: DEFINING THE RESOLUTION OF A NETWORK FOR TRANSPORTATION ANALYSES: A NEW METHOD TO IMPROVE TRANSPORTATION PLANNING DECISIONS

Yuchen Cui, Doctor of Philosophy, 2016

Dissertation Directed by: Professor Marie Howland
School of Architecture, Planning, and Preservation
University of Maryland

Professor Rolf Moeckel
Department of Civil, Geo and Environmental Engineering
Technical University Munich

Travel demand models are important tools used in the analysis of transportation plans, projects, and policies. The modeling results are useful for transportation planners making transportation decisions and for policy makers developing transportation policies. Defining the level of detail (i.e., the number of roads) of the transport network in consistency with the travel demand model's zone system is crucial to the accuracy of modeling results. However, travel demand modelers have not had tools to determine how much detail is needed in a transport network for a travel demand model. This dissertation seeks to fill this knowledge gap by (1) providing methodology to define an appropriate level of detail for a transport network in a given travel demand model; (2) implementing

this methodology in a travel demand model in the Baltimore area; and (3) identifying how this methodology improves the modeling accuracy.

All analyses identify the spatial resolution of the transport network has great impacts on the modeling results. For example, when compared to the observed traffic data, a very detailed network underestimates traffic congestion in the Baltimore area, while a network developed by this dissertation provides a more accurate modeling result of the traffic conditions. Through the evaluation of the impacts a new transportation project has on both networks, the differences in their analysis results point out the importance of having an appropriate level of network detail for making improved planning decisions.

The results corroborate a suggested guideline concerning the development of a transport network in consistency with the travel demand model's zone system. To conclude this dissertation, limitations are identified in data sources and methodology, based on which a plan of future studies is laid out.

DEFINING THE RESOLUTION OF A NETWORK FOR TRANSPORTATION ANALYSES: A NEW METHOD TO IMPROVE TRANSPORTATION PLANNING DECISIONS

by

Yuchen Cui

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
2016

Advisory Committee:
Professor Marie Howland, Co-chair
Professor Rolf Moeckel, Co-chair
Professor Gerrit Knaap
Dr. Frederick Ducca
Professor Paul Schonfeld

DEDICATION

To my grandmother and grandfather

Thank you.

ACKNOWLEDGEMENTS

I am deeply grateful to my advisor Rolf Moeckel for his guidance and support throughout my dissertation research as well as many project collaborations. His style of doing modeling and research – simple, elegant, and efficient, has greatly influenced my own. It has been an honor to work with him and get to know him as a mentor and a friend.

I am also deeply grateful to my advisor Marie Howland for her guidance, advice and encouragement throughout my Ph.D. study at the University of Maryland. Marie has always responded with a smile to my knocks on her office door. Her support has been critical to my dissertation research and career choices.

I am grateful to my dissertation committee, Gerrit Knapp, Frederick Ducca and Paul Schonfeld, for their constructive comments and suggestions to improve this work. I am also grateful to Howie Baum who taught me planning theory and provided many stimulating conversations on and beyond planning. Finally, my time at the UMD would not have been enjoyable without my friends, Vanessa Leon, Zhi Li, and Basheer Saeed.

I am grateful for the financial support from the National Center for Smart Growth Research and Education and a dissertation grant from the University of Maryland.

I want to thank my parents for their nurturing and their trust on all of my decisions. My life would not have been the same without my husband, Xiao, whose love and faith has made me a complete person.

I could not possibly be writing this dissertation if there had not been my maternal grandmother and late grandfather, who devoted their lives to making me an educated and strong person. This dissertation is dedicated to them.

TABLE OF CONTENTS

LIST OF TABLES vi

LIST OF FIGURES vii

1 INTRODUCTION 1
1.1 Motivation 1
1.2 Objective 3

2 STATE OF THE ART 5
2.1 Consistency between the Network and Zone Systems 5
2.2 Define Network Resolution 11
2.3 Summary of the Review 14

3 DATA 15
3.1 Maryland Statewide Transportation Model 15
3.1.1 Model Structure 15
3.1.2 Area Type 17
3.1.3 Traffic Analysis Zones (TAZs) 18
3.2 Highway Network Data 19
3.2.1 Data Sources 19
3.2.2 Modification on Centerline Network 22
3.2.3 Link Attributes 26
3.3 Validation Data 33

4 METHODOLOGY 35
4.1 A Comprehensive Model to Define Network Resolution 35
4.2 Study Area 37
4.2.1 Subarea Analysis 37
4.2.2 Post-processing of Subarea Analysis 40
4.2.3 Finest Highway Network 41
4.2.4 Modification of Finest Network 43
4.3 Trip Assignment 44
4.3.1 Trip Assignment Method – All-or-nothing Assignment 44
4.3.2 Iterative Trip Matrix Loading Method – Method of Successive Averages 45
4.3.3 Path-tracing between Origins and Destinations 47
4.3.4 Model Convergence Criteria 49
4.3.5 Computational Efficiency 52
4.3.6 Performance Measures 52
4.4 Identify Irrelevant Links 55

5 RESULTS AND ANALYSES 59
5.1 Results of Subarea Analysis 59
5.2 Adjustment of Computation Time 59
5.2.1 Convergence Level 60
5.2.2 Zero-volume Links on Finest Network 62

5.2.3 Modified Finest Network 65
5.3 Identify Irrelevant Links 66
5.3.1 Sort Origin-Destination Trip Table 67
5.3.2 Path-tracing Model 68
5.3.3 Identify and Remove Irrelevant Links 70
5.4 Network-defining Models 72
5.4.1 Results of a Provisional Model 72
5.4.2 Zero-volume Links after Removing Irrelevant Links 75
5.4.3 Smaller Steps of Irrelevant Link Removal 77
5.4.4 Larger Coverage of OD Pairs 78
5.5 Analyses of Finest Network and Final Network 81
5.5.1 Comparison of Modeled Volumes – Individual Links 81
5.5.2 Comparison of Modeled Volumes – Entire Network 83
5.5.3 Comparison of Validation Results 85
5.5.4 A Case Study on Final and Finest Networks 86
5.6 Implementation of a New Transportation Project 88

6 CONCLUSIONS AND IMPLICATIONS 93
6.1 Summary of Research and Findings 93
6.2 Implications for the Development of Travel Demand Models 96
6.2.1 Develop a New Network 98
6.2.2 Calibrate an Existing Network 99
6.3 Future Direction 100

APPENDIX A – MODEL DIRECTORY 102

APPENDIX B – EXECUTABLE SCRIPTS AND CODES 104

GLOSSARY 144

REFERENCES 148

LIST OF TABLES

Table 3.1 Area Type Look-up Table Defined by the BMC 17
Table 3.2 Digital Street Files Provided by the SHA 19
Table 3.3 Limitations of Three Networks 22
Table 3.4 Definition of Functional Class 27
Table 3.5 Look-up Table of Free Flow Speed Modified from the BMC Definition 30
Table 3.6 Look-up Table of Link Capacity Modified from the BMC Definition 31
Table 3.7 Traffic Count Stations 33
Table 4.1 Example of Identification of Irrelevant Links – Trip Table 57
Table 4.2 Example of Identification of Irrelevant Links – Trip Share Calculation 58
Table 4.3 Example of Identification of Irrelevant Links – Trip Share Results 58
Table 5.1 MSA Convergence Progression Based on Finest Network 60
Table 5.2 Validation Results of Different Convergence Levels Applied to Finest Network 61
Table 5.3 Links with Zero Volume in the PM Assignment on Finest Network 63
Table 5.4 Comparison of Validation Results of the Fines Network and Its Modification 65
Table 5.5 Distribution of Trip Demand Percentage 67
Table 5.6 Path-tracing Result for Origin Zone 64 and Destination Zone 79 69
Table 5.7 Link-volume Matrix Holding Path-tracing Results 70
Table 5.8 Trip-share Matrix 71
Table 5.9 Parameters and Validation Analysis of the Provisional Model 74
Table 5.10 Parameters and Validation Analysis of the Smaller-Step-Model 77
Table 5.11 Parameters and Validation Analysis of the Larger-Coverage-Model 79
Table 5.12 Parameter Testing on Last Iteration on Network V5 80
Table 5.13 VMT and VHT on Finest and Final Networks 84
Table 5.14 Volume-count Agreement on Finest and Final Networks 85

LIST OF FIGURES

Figure 1.1 Different spatial relationships between the network and zone systems 3
Figure 3.1 Overview of MSTM model structure (source: MSTM User's Guide) 16
Figure 3.2 Zone system and area type distribution in the study area 18
Figure 3.3 Digital networks provided by the SHA showing spatial representations 20
Figure 3.4 Differences in spatial presentation of three networks 21
Figure 3.5 Missing links in Baltimore City 23
Figure 3.6 Misrepresentation of divided highway links 24
Figure 3.7 Example of disconnected nodes 26
Figure 3.8 Link attribute - functional class in the Baltimore area 28
Figure 3.9 Link attribute - number of lanes in the Baltimore area 29
Figure 3.10 Traffic count stations in the Baltimore area 34
Figure 4.1 A comprehensive model to define network resolution 36
Figure 4.2 MSTM network and study area boundary displayed in Cube 38
Figure 4.3 Subarea network extraction renumbering in Cube 38
Figure 4.4 MSTM subarea network of the study area 39
Figure 4.5 Zone system defined for the study area 41
Figure 4.6 Finest highway network with centroid connectors generated from zone centroids 43
Figure 4.7 Path-tracing method applied in trip assignment 48
Figure 4.8 Example of trip matrix transformation and sorting procedure 49
Figure 4.9 Example of identification of irrelevant links – trip assignment 57
Figure 5.1 Trip assignment results on the finest network 63
Figure 5.2 Trip assignment results on the modified finest network 66
Figure 5.3 Procedure of sorting OD trip interchange 68
Figure 5.4 Trip assignment result and path-tracing result for the 1st largest OD pair 69
Figure 5.5 Decision-making process of identifying the ideal network resolution 73
Figure 5.6 Example of zero-volume links on network v3 76
Figure 5.7 Example of a dangling zero-volume link on network v6 77
Figure 5.8 Link flow rate distributions of interstate highway links 82
Figure 5.9 Link flow rate distributions of freeway and expressway links 83
Figure 5.10 Comparisons between the final and finest networks for zone 62 88
Figure 5.11 The location and extent of I-695 new project in the fines network 89
Figure 5.12 V/C ratio changes on finest network after implementing the new project 90
Figure 5.13 V/C ratio changes on final network after implementing the new project 91

1 INTRODUCTION

1.1 Motivation

A travel demand model is a powerful transportation planning tool that provides planners with estimates of traffic congestion and public transit ridership, which can further help provide estimates of infrastructure requirements, air quality, water quality, and demand of dwelling units through connections with other models. Travel demand models can be used to test the impacts of alternative planning policies or infrastructure projects before policies or projects are implemented.

The development of a travel demand model begins with the formation of a traffic analysis zone (TAZ) system to represent travellers' demand and a transport network system to represent network supply. The spatial resolution of the TAZ system depends on the types of information the travel demand model needs to provide for planning analyses, the characteristics of the region being modeled, and the availability of geographic and socio-economic data associated with the TAZs. As part of the modeling process, defining the transport network's level of detail (e.g. number of roads) in consistency with the level of detail of the TAZ is critical to the modeling accuracy of a travel demand model. In other words, the amount of detail in the transport network must be in proportion to the amount of travel demand to be placed on the transport network.

For instance, if a travel demand model accounts for a greater spatial resolution with a detailed TAZ system, where each zone covers a relatively small land area, its transport network should incorporate the collectors and local roads. If transportation planners want to understand regional travel patterns (on interstate highways and major

arterials) and develop a statewide travel demand model, its TAZ system should be more aggregated and therefore, lower class roads are usually removed from the transport network.

An inconsistent spatial resolution between the network and zone systems will result in inaccurate modeling results. However, travel demand modelers frequently overlook the importance of the consistency in the level of detail between the network and zone systems *(1-3)*. If the network system has greater level of detail than the zone system, congestion is likely to be underestimated. The reason is that some trips become intrazonal trips given a comparatively coarse zone system. Since the trip assignment does not take intrazonal trips into account, a network with too much detail compared to the zone system will underrepresent congestion. If the network is over-simplified for a given zone system, congestion will be overestimated as too many trips are loaded on a transport network without the capacity to absorb them. Examples in Figure 1.1 help illustrate how the spatial relationship between the network and zone systems influences modeling results.

As depicted in Figure 1.1 (a), a coarser zone system and a finer network will underestimate traffic congestion. The reason is that a highly aggregated zone system produces more intrazonal trips that are not modeled by trip assignment, and fewer interzonal trips must distribute on many more network links. Figure 1.1 (b) shows that a finer zone system and a coarser network will overestimate traffic congestion. The reason is that trips between zone pairs in reality would take different routes; however, there are fewer links coded on the network, and therefore all these trips are forced onto the same route. To date, there is no tools to define how much detail is needed for a transport

network so that it is spatially consistent with a given zone system, as presented in Figure 1.1 (c).

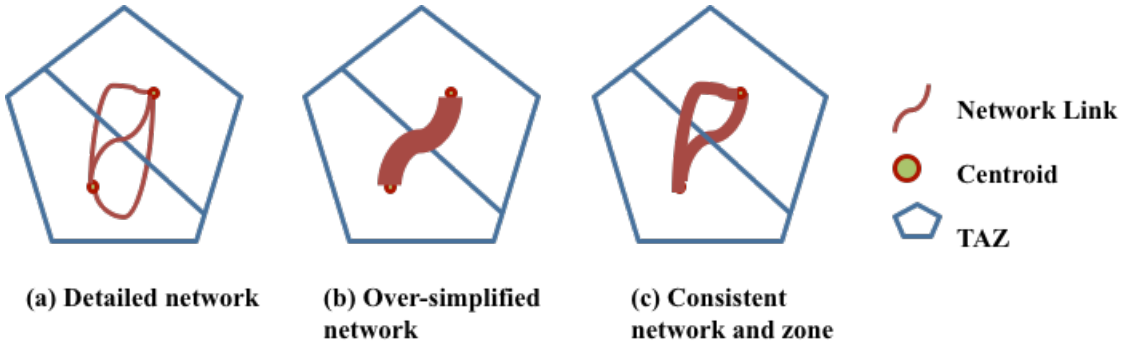

Figure 1.1 Different spatial relationships between the network and zone systems

1.2 Objective

The objective of this dissertation is to develop a network-defining model and to identify an ideal level of network detail for a given zone system in a travel demand model. The specific research goals are the following:

- For a given zone system, develop measures and tools to assess the performance of a specific network system and the zone-network spatial consistency.
- For a given zone system, develop a method to iteratively refine the network resolution until it is spatially consistent with the zone system.
- Present methodology and a set of guidelines on how to define the network resolution for a given zone system.
- Use a calibrated travel demand model to implement the network-defining model in the Baltimore area (Baltimore County and Baltimore city), MD.

The criterion used to determine the ideal network is based on the comparison between modeled traffic volumes and observed traffic counts. It can be expected that the research

methodology proposed in this dissertation is easy to implement in other travel demand models. The ultimate goal of this dissertation is providing more accurate modeling results to transportation planners and policy makers, and thereby, to improve transportation planning decisions.

This dissertation begins with a review of previous research studies underlying the study objective. In Chapter 3, the data sources and modeling platform are introduced. Chapter 4 presents the research methodology, including study area, methods and algorithms involved in the development of the network-defining model. The developed networks and analysis results of network performance are presented in Chapter 5. The dissertation concludes with a summary of research and findings, recommendations for the development of travel demand models, and direction of future research.

2 STATE OF THE ART

In this chapter, an overview of past research is presented and discussed. First, the literature review surveys existing findings on how different zone and network structures influence the modeling results. Specifically, the measures used in these studies to evaluate the network performances are reviewed in detail. Second, some studies on network aggregation are presented; and the review is focused on the criteria applied to aggregating a detailed network into a less detailed one. The search for literature has not located any study on how to define a network system in consistency with the zone system in terms of their spatial resolutions.

2.1 Consistency between the Network and Zone Systems

Jansen and Bovy *(4, 5)* defined the spatial resolutions of the network and zone systems concurrently, and investigated the effects of different spatial resolutions on the modeling results of a travel demand model. To be specific, they defined the network system in three levels of detail – fine, medium, and coarse by a reduction method. In the fine-level model, the network was almost identical to the actual road network; the medium-level model included all the arterial and collector roads; and the coarse-level model only represented the arterial roads. Once the network system was defined, the zone system was formed by the network links: the TAZs were the "holes" delimited by the network links. As a result, the zone boundaries lined up with the network links. The authors experimented three models using two assignment methods – all-or-nothing and equilibrium, in a travel demand model developed for the Dutch city of Eindhoven.

In analyzing the impacts of different levels of spatial detail on the modeling results, the authors examined the frequency distributions of the link loads by three link groups (primary, secondary and local roads) and compared the modeled volumes with observed traffic counts for the fine-, medium-, and coarse-level models. They also compared the analysis results of two different trip assignments. Results can be summarized as follows: (1) for network-wide total modeled volumes, differences were small between three models with different levels of detail; (2) the modeled volumes on primary links were almost equal in the fine- and medium-level models but substantially higher in the coarse-level model; (3) distributions of link loads in the fine- and medium-level models were almost equal, whereas the distribution in the coarse-level model was biased; (4) the modeled volumes in the fine-level model were the closest to the observed traffic data; and (5) the equilibrium assignment outperformed the all-or-nothing assignment in all three models.

Jansen and Bovy's work is the first to prove that the level of spatial detail (i.e. zone size and network resolution) of a travel demand model has significant impacts on the modeling results. In particular, they strategically defined the fine-, medium-, and coarse-level models and presented incremental improvement as the model's level of spatial detail increased. More importantly, Jansen and Bovy identified the sources of the modeling errors: systematic error and random error. When downgrading the model from a finer level (e.g. actual road network) to a coarser level (e.g. simplified network), the assignment forced the same amount of trips through a smaller network, which introduced the systematic error (bias); during the same process, there was also a change in routing possibilities and redistributing volumes between routes due to network changes and

equilibration, which introduced the random error (dispersion around the bias). In Jansen and Bovy's empirical study, the network reduction from the fine-level model to the medium-level model led to a significant increase in modeled volumes on primary links, which was the systematic error; the equilibration process (all-or-nothing or equilibrium) redistributed traffic volumes on the network, which led to the dispersion around the systematic error.

Long and Stover *(1)* conducted an empirical study exploring the impacts on traffic assignment results of three network systems differing in spatial detail. The authors selected the Waco, TX urban area for the study and defined three street systems with different degrees of spatial detail: a normal-detail network, an intermediate network, and a detailed network. The normal-detail network was a coarse network that only had higher class links; the detailed network contained all existing streets; and the spatial detail of the intermediate network was between the other two networks. In this study, all-or-nothing trip assignment was applied to the three networks.

For screenline crossings, arterial streets and selected links, modeled volumes were validated against observed traffic counts using root mean square error (RMSE). Validation results of the three networks all showed some differences between traffic counts and modeled volumes. In particular, the validation results on arterial and major collector roads showed no improvement as the network detail increased; at the same time, the computation time increased dramatically.

It is worth mentioning that the authors discussed the relationship between zone size and network detail. For the detailed network, the authors constructed a zone system constituted by city blocks; and for less detailed networks, small zones (city blocks) were

aggregated to create a coarser zone system. In less detailed models, the authors found out that trips that would be assigned to local streets in the detailed model had become intrazonal and were not assigned on the present network. Furthermore, short trips that would use local streets in the detailed model had to be rerouted to a higher class route.

Khatib, Chang, and Ou *(2, 6)* conducted a simulation study using the Idaho statewide travel demand model to investigate how different zone structures and network details affected modeling results. They developed 11 zone structures and two network systems in different levels of spatial detail. The fine-level network system included interstate, principal arterial, minor arterial, and major collector roads in both urban and rural areas, and minor collector roads in rural areas; and the coarse-level network included interstate, US, and state highway roads. The 11 zone structures varied in zone size (county, census tract, and census block group) and centroid location.

The authors examined the differences in model performance by comparing trip length (average travel time) and percentage of interzonal trips between different models. They also compared the modeled volumes with observed traffic counts by three measures: ratio of modeled volumes over observed traffic counts, correlation between modeled volumes and observed traffic counts, and percent root mean square error (PRMSE) between modeled volumes and observed traffic counts. Results showed that the level of network detail had little effect on modeled trip length and percentage of interzonal trips. Regardless of zone structures, the PRMSE was higher on a less detailed network; and a model with a larger zone size and a less detailed network always had a lower PRMSE value.

The most important contribution of this study is that it analyzed different combinations of zone structures and network systems. The results suggested that the zone structure and network system should always be consistent in their levels of spatial detail. Another highlight from this study is that the influence of network detail on trip length was small. This finding was consistent with Jansen and Bovy's analysis of distributions of link loads on the fine- and medium-level models. The examination of the PRMSE values suggested that this indicator was more sensitive to the change of network detail than the change of zone structure.

Jeon et al. *(7, 8)* conducted an empirical study to analyze the impacts of different zone structures and network systems on modeling results using a travel demand model developed for Seoul City in Korea. Similar to Bovy and Jansen's study, they defined three levels of network detail: the fine-level model represented the road network in reality; the medium-level model included expressway, major arterial, and minor arterial roads; and the coarse-level model included expressway and major arterial roads. The zone system was adapted to the network system: they were the "holes" formed by the network so that the zone boundaries lined up with network links.

When analyzing these models with different levels of spatial detail, the authors applied several performance measures, including the PRMSE and correlation coefficient between modeled volumes and observed traffic counts. They also compared the modeled results using derived socio-economic costs, which included vehicle operating costs, travel time costs, environmental costs, and vehicle accident costs. These derived costs were calculated by using travel speed, travel time, link length, and traffic volume.

Comparing modeled volumes between the fine-, medium-, and coarse-level models, they found that the reduction in network detail led to increased intrazonal trips and rerouting effect. They also evaluated interaction effects between different levels of network detail and zone sizes. Consistent with Khatib, Chang, and Ou's findings, this study emphasized the importance of having consistent zone and network systems in their levels of spatial detail to achieve more accurate modeling results.

In summary, these studies conducted empirical research to evaluate how different levels of network detail and zone structures affected the modeling results in a travel demand model. Jansen and Bovy, Long and Stover and Jeon et al. developed the network system first, then manually developed the zone structure according to the network detail; while Khatib, Chang, and Ou developed the network system and the zone system separately. Jansen and Bovy's, Long and Stover's and Jeon et al's studies compared models with consistent zone and network systems at different degrees of spatial resolution (fine-, medium-, and coarse-level). Findings from Jansen and Bovy's and Jeon et al's studies suggested that higher spatial resolution would yield more accurate modeling results, while Long and Stover's findings suggested no improvement as the model's degree of spatial detail increased. Khatib, Chang, and Ou's as well as Jeon et al's studies conducted performance comparison between models with inconsistent zone and network systems and models with consistent zone and network systems. Their findings confirmed the importance of applying consistent zone and network systems in a travel demand model to achieve more accurate modeling results.

2.2 Define Network Resolution

To date, there has been no research conducted in defining the network resolution in consistency with the zone system in a travel demand model. However, the longer computation time associated with large size networks has motivated researchers to simplify the network by extracting important links from a detailed network, or to solve the network equilibrium problems on an aggregated version of the detailed network. These studies proposed heuristic or theoretical methods in seeking less computationally demanding solutions while preserving the modeling results of the original network. There is a lack of consideration of the spatial relationship between the network and zone systems in these studies. However, the network aggregation strategies, such as link removal or link combination, as well as the parameters applied to defining the network aggregation level or reduction scale, such as link flow under network equilibrium, should shed some light on the research methodology of this dissertation.

Haghani and Daskin *(9)* provided a heuristic approach to extract a sub-network from a detailed network. Their research was motivated to reduce computer storage and computation time when solving network-related problems. The basic assumption of their methodology is that, when traffic flows distribute over a network following user equilibrium, there will be some links not carrying a significant amount of traffic volumes; therefore, removing these insignificant links can reduce computational cost while the modeling results on the original network can be reproduced on an aggregated network. When the original network reached its equilibrium, the authors sequentially identified and deleted one insignificant link at a time, if this link was below a certain percent of the maximum equilibrium link flow on the network. They updated the network and trip

matrix and provided modeling results after each model iteration until no better sub-network can be found. Through the implementation in a network design problem, the authors found that users' travel time was overestimated on an aggregated network compared to that was measured on the detailed network.

Ruddell and Raith *(10)* proposed a zonal-based aggregation method to find the equilibrated assignment solution for the original network. This aggregation method first grouped some zones together using a certain criterion, and then assigned trips to the network using all-or-nothing assignment algorithm. After that, they applied a path equilibration algorithm to reassign the all-or-nothing results on the shortest route(s) between each origin-destination (OD) pair to achieve an equilibrium solution on the aggregated network. For zones combined together, shortest paths were found using all-or-nothing assignment and were later chained together with the assignment solution found on the aggregated zone system. Finally, the equilibrated paths were transferred to the original network. Through its implementation using different networks, the authors concluded that this method could significantly shorten computation time to find a path-based equilibrated solution for a given network.

Connors and Watling *(11)* presented an analytical framework for the network aggregation problem. They considered the path-based costs as a function of commuters' demand between OD pairs, and applied a linear cost-flow function to predict approximate flows on the aggregated network. They implemented this method in a pilot study, which had a highly aggregated two-link network and a detailed network. Their results (modeled flows on the two links) revealed that by depending link flows on OD demand, the predicted flows were very similar to the user-equilibrium solution computed on the

detailed network. Even though the authors proved their algorithm could reproduce similar traffic flows using an aggregated network, they did not provide a complete methodology on how to define the aggregate network for a given detailed network.

Chan *(12)* presented a network aggregation technique to reproduce the trip assignment results of a detailed network on an aggregated sub-network. The author first categorized network links into five groups: the access links delivering flows to its destination, the egress links carrying flows from its origin, the bypass links facilitating turning movements, the line-haul links directly connecting two zones, and the intra links carrying either intra-inter flows or only intra flows. The network aggregation followed three steps for each OD pair: first, links were grouped into five groups as mentioned above; secondly, travel times were summed up for links belonging to the same group; finally, the weighted average travel time (use individual link volumes as weights) was calculated as the travel time for each link group. The author compared the aggregated network and detailed network in terms of their travel times. It was found that by using an aggregated network could significantly reduced computation time to find a comparable network assignment solution.

In summary, these studies aim to reproduce the equilibrium assignment results of the detailed network on a less detailed network as a means to reduce computational complexity. Haghani and Daskin proposed a link removal method, which identified and removed an insignificant link from the detailed network per iteration. Their method terminated the program when no better sub-network can be found. Both Chan's and Ruddell and Raith's studies designed an aggregation method to approximate the equilibrium solutions of the original network. Chan combined network links with similar

functions for each OD pair; Ruddell and Raith applied a different aggregation approach by combining adjacent zones, and then transferred the equilibrium solution found on the aggregated zone system to the original model. Connors and Watling utilized the demand-cost functions and cost-flow functions to establish a linear relationship between OD demand and equilibrium flow on a certain link. Even though these articles are different from this dissertation in their research objectives, they provided an important clue to start from a detailed network, and approach the optimal network incrementally.

2.3 Summary of the Review

To summarize, several highlights can be identified from this review. First, empirical evidence has been found regarding the effects of the zone-network spatial consistency on the modeling accuracy of a travel demand model. Second, a variety of performance measures have been used to validate the modeling results against observed traffic counts. Third, on a detailed zone system, having a less detail network would lead to an overestimation of traffic congestion, especially on higher class roads. Fourth, there has been no solution provided on how to define the most appropriate level of network detail given a zone system in a travel demand model. Finally, it would be promising to start defining network resolution from a detailed network and incrementally refine it based on certain rules.

3 DATA

The development of a detailed network, implementation of methodology, and validation analysis of modeling results require different data sources. This chapter describes the data gathered for these functions. The Maryland State Highway Administration (SHA) provided the data for network development. The implementation of this study is based on the Maryland Statewide Transportation Model. The data used in validation analysis was also provided by the SHA.

3.1 Maryland Statewide Transportation Model

The Maryland Statewide Transportation Model (MSTM) is a trip-based statewide travel demand model. By design, the MSTM is a multi-layer model working at the regional, statewide and urban levels. Its first layer represents national travel and freight patterns. The second layer is a multi-state layer, including Maryland, Washington DC, Delaware and selected areas in Pennsylvania, Virginia and West Virginia. The third layer is an urban layer covered by two models developed by local Metropolitan Planning Organizations (MPOs), and it is only used for model reconciliation purpose (see green box in Figure 3.1) but not in the MSTM production model runs *(13)*. The layer-based approach allows for better representation of multiple trip types including short distance trips and long-distance trips, as well as multiple travel modes, such as urban transit and regional commercial vehicles.

3.1.1 Model Structure

Figure 3.1 summarizes the MSTM components within the multi-state and national layers. On the person travel side, it includes a long distance travel model for person trips longer

than 50 miles, accounting for through trips with at least one trip end within Maryland. This component also includes a multi-state short distance travel model for person trips classified by study area residents. The person travel model follows a three-step sequence, including components of trip generation, destination choice (trip distribution), and mode choice. On the freight side, it includes a commodity-flow based long-distance freight model for truck trips that are longer than 50 miles.

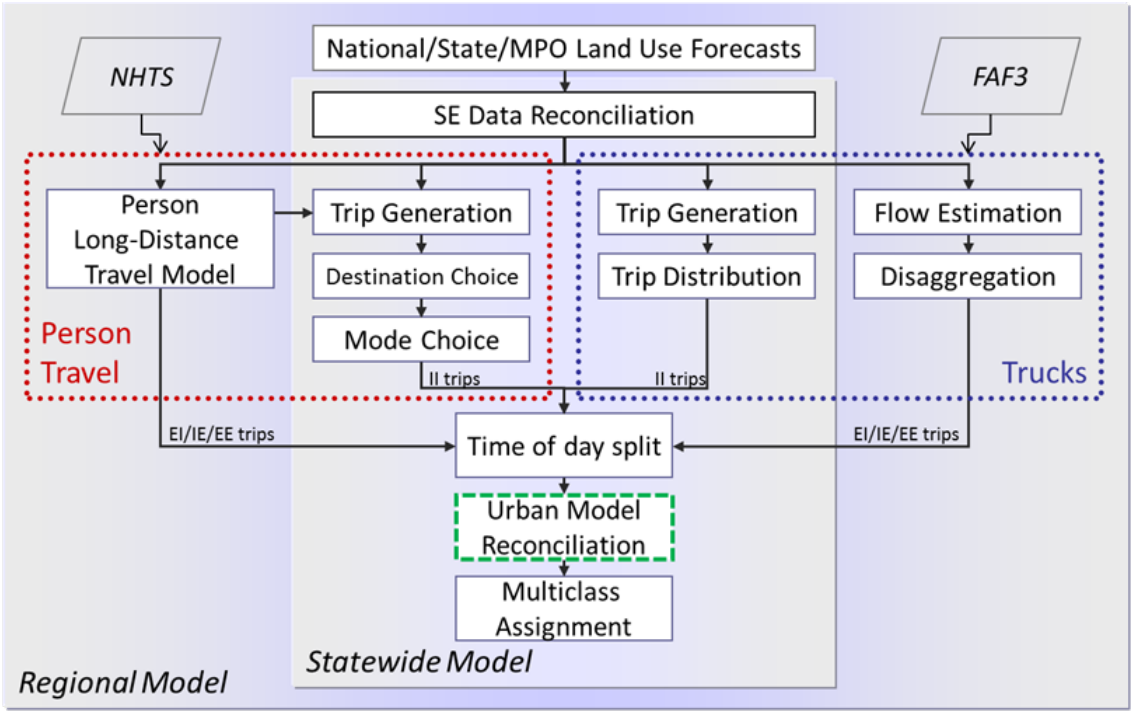

Figure 3.1 Overview of MSTM structure (source: MSTM User's Guide)

The outputs from the person travel side are 18 trip tables, including home-based work trips for five income groups, home-based shopping trips for five income groups, home-based other trips for five income groups, home-based school trips, non-home-based work trips, and non-home-based other trips; and one trip table representing long-distance person trips. The output from the truck side is one trip table representing short-distance and long-distance truck trips.

In total, there are 20 trip tables covering national and statewide travel patterns. These outputs are important inputs for the Subarea Analysis, which will be discussed in detail in Section 4.2.1.

3.1.2 Area Type

Area type is a measure to classify land use intensity, based on population and employment density. A higher area type value indicates more intersections, driveways, traffic signals, and turning movements, as well as lower capacity and travel speed in that area. In this dissertation, the area type definition was provided by the Baltimore Metropolitan Council (BMC) *(14)*. Table 3.1 presents the area type look-up table. The area type is an important variable to determine network attributes, including free flow speed and link capacity.

Table 3.1 Area Type Look-up Table Defined by the BMC

Employment/Acre	**Households/Acre**									
	< 0.5	**0.5 - 1.0**	**1.0 - 1.5**	**1.5 - 2.25**	**2.25 - 3.0**	**3.0 - 4.0**	**4.0 - 5.0**	**5.0 - 7.5**	**7.5 - 11**	**> 11**
< 1.5	1	1	2	2	3	3	4	5	5	6
1.5 - 3.5	1	1	2	2	3	3	4	6	6	6
3.5 - 6.5	1	1	2	2	3	3	4	6	6	6
6.5 - 12	1	2	2	3	3	4	4	6	6	7
12 - 20	1	2	3	3	4	4	5	7	7	7
20 - 30	2	3	4	4	5	5	5	7	7	7
30 - 45	3	4	4	5	5	6	6	7	7	8
45 - 70	3	4	4	5	5	6	7	8	8	8
70 - 110	4	4	5	6	6	7	8	9	9	9
> 110	4	5	6	7	7	8	9	9	9	9

3.1.3 Traffic Analysis Zones (TAZs)

The TAZs and their centroids were defined at the beginning of the MSTM development. The TAZ delineation in the MSTM followed certain guidelines *(13)*.

The MSTM has 1,588 zones compassing entire State of Maryland, State of Delaware, and District of Columbia, and selected areas in the States of Virginia, West Virginia, and Pennsylvania. This dissertation utilizes the zone system defined by the MSTM, but in a smaller study area within the Baltimore County and Baltimore City including 362 TAZs. The area type of this zone system is defined according to Table 3.1. Figure 3.2 shows the MSTM-defined zone system in the dissertation's study area, and its area type distribution. Later in this chapter, the zone system along with its area type attribute will be used in defining network attributes.

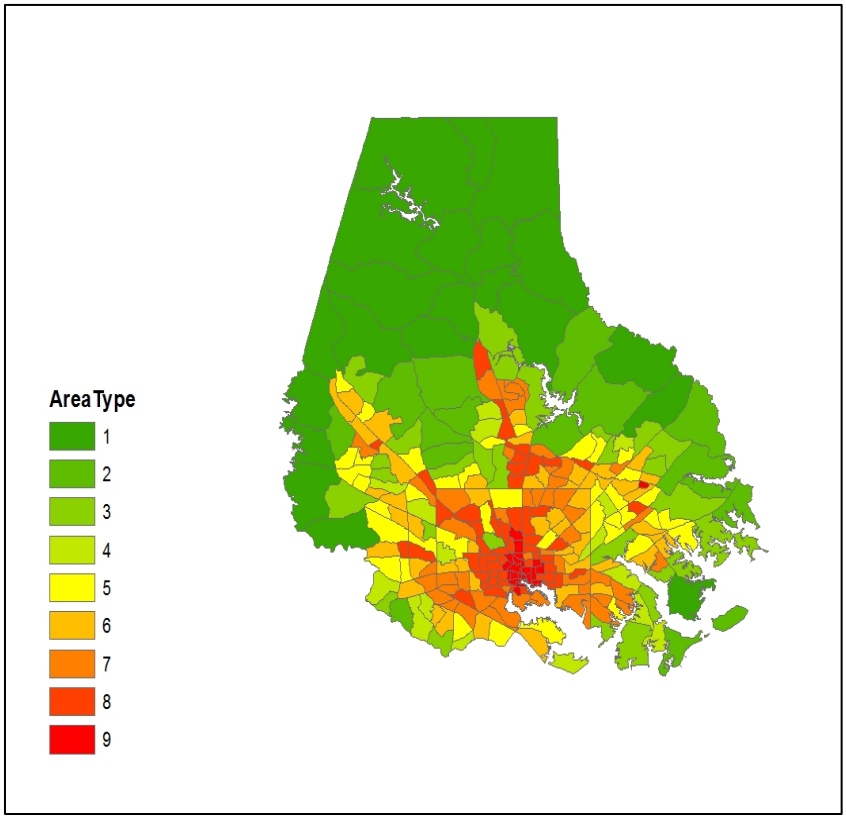

Figure 3.2 Zone system and area type distribution in the study area

To make this research manageable and reduce the computation time, zones with similar area type are aggregated. This procedure leads to a coarser zone system with 81 TAZs, including eight external zones. Section 4.2.2 provides more details on the aggregation process.

3.2 Highway Network Data

The highway network represents the transport system in a travel demand model. Attributes on a highway links represent the level of service on that segment. A correct representation of the highway network is crucial to achieving accurate modeling results. In this section, the sources for network data, definition of network attributes, and modifications made during the network development are discussed.

3.2.1 Data Sources

The spatial resolution of the MSTM network is not detailed enough as required by the dissertation methodology. Digital street files were requested from the SHA for the development of a detailed network system. Table 3.2 summarizes the digital files provided by the SHA.

Table 3.2 Digital Street Files Provided by the SHA

File Name	Date Received	Format	Description
BaltimoreCounty&City_CL	05/2013	ArcGIS link shapefile	Centerline network that contains true shape of road links in MD
Functional_Class_MD	06/2014	ArcGIS link shapefile	Contains selected road links with functional class in MD
Lanes_Speed_Limit_MD	06/2014	ArcGIS link shapefile	Contains selected road links with lane numbers and speed limit in MD

In order to create a network that represents the highway system serving the study area, the Centerline network is used as the base network. Link attributes on every other network will be joined to the Centerline network. Figure 3.3 presents the three networks showing their spatial representations of the Baltimore area.

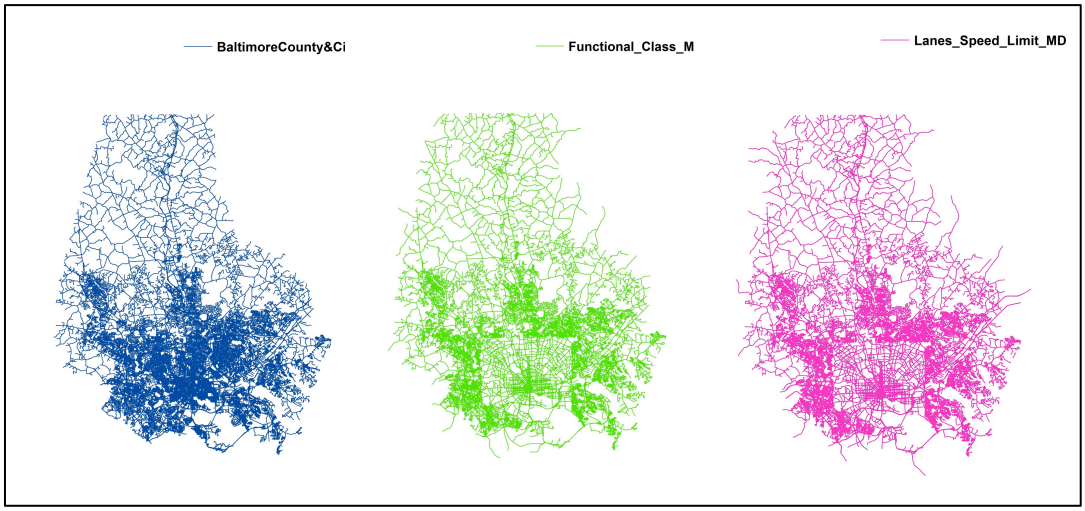

Figure 3.3 Digital networks provided by the SHA showing spatial representations

It can be noticed that the Centerline network represents a network system with more detail, compared to the lane/speed limit network and functional class network. While the three networks are consistent in geographic projection, the lane/speed limit and functional class networks have less spatial resolution in the Baltimore City. When taking a closer look at the three networks on top of each other as shown in Figure 3.4, it is obvious that they are not consistent in their spatial presentation, especially at interchanges. For instance, the lane/speed limit network does not distinguish divided highways (pointed by arrow a); the functional class network has fragmented links at some locations (pointed by arrow b); and the Centerline network has segmented links at interchanges (pointed by arrow c), bridges, and tunnels where these links should not be intersected by other links.

Figure 3.4 Differences in spatial presentation of three networks

To better understand the differences of the three networks, Table 3.3 summaries their advantages and compares their limitations. For the sake of this dissertation, it requires a detailed highway network that represents the highway system in reality. Ideally, the detailed network should have correct attributes: speed limit, functional class, capacity, and toll rate; and correct spatial representations: directionality of divided roads and one-

way roads, and connectivity. Since none of the three networks meets these requirements, several modifications will be undertaken to develop the detailed network.

Table 3.3 Limitations of Three Networks

File Name	Limitations	Advantages
BaltimoreCounty&City_CL	Segmented links at important locations, including bridges, tunnels, and interchanges; link direction not indicated	True shape of road links
Functional_Class_MD	Fragmented links; link direction not indicated; divided highway links not distinguished	Contains function class on links
Lanes_Speed_Limit_MD	Link direction not indicated	Contains speed limit and number of lanes on most links

3.2.2 Modification on Centerline Network

Since the Centerline network (BaltimoreCounty&City_CL) is supported by the ongoing Centerline Project in the SHA, it is supposed to have the best spatial resolution and true to shape. Other network attributes, including the functional class, number of lane, and speed limit, will be joined onto the Centerline network. Based on the Centerline network, efforts are made to reconcile the spatial inconsistencies of these networks.

3.2.2.1 Missing Attributes in Baltimore City

Figure 3.5 is a zoom-in map of the Baltimore City. Because of the missing links on the lane/speed limit and functional class networks, the Centerline network does not have any attribute defined on these links. Eventually, the number of lanes, speed limit, and functional class were manually added to these links on the Centerline network.

Figure 3.5 Missing links in Baltimore City

3.2.2.2 Missing Attributes on Divided Highways

As indicated in Table 3.2, the functional class network does not distinguish divided highway links. For instance, a one-way divided highway link is shown as a two-way undivided link. As a result, on divided highways of the Centerline network, only links in one direction have functional class defined, as presented in Figure 3.6. In ArcGIS, the

Transfer Attributes function was used to add missing attribute to links in the other direction. The *transfer distance* is 80 feet.

A search of zero-value attribute was conducted in ArcGIS to add missing attributes on the Centerline network. After this step, every link on the Centerline network has number of lanes, speed limit, and functional class coded in their network attributes.

Figure 3.6 Misrepresentation of divided highway links

3.2.2.3 Connectivity on Special Highway Segments

Another shortcoming of the Centerline network is its segmented links at bridge and tunnel crossings, and at some interchanges. Links at these locations appear to intersect on

a two-dimensional map, but in fact one link passes under the other one(s). A network with segmented bridges, tunnels and interchanges cannot be used in trip assignment because trips will be assigned on routes that do not exist in reality.

In ArcGIS, the National Bridge/Tunnel Inventory network was used to fix this problem. A considerable amount of manual work was spent on connecting segmented bridge and tunnel links and interchanges. After checking the connectivity of the Centerline network, it assures that this network can be used in trip assignment.

3.2.2.4 Ramp Indication

By this step, the Centerline network has been imported into Cube, and will be prepared for following analyses. Highway ramps were identified with "RP" in their network attribute and were imported as one-way links in Cube. The directionality and connectivity of ramp links were checked manually in Cube to make sure that all ramps are correctly connected to highway exits and entrances.

3.2.2.5 Directionality of Divided Highways

Since the ArcGIS is unable to identify directionality, all links (except ramp links) were imported in Cube as two-way links. However, this approach creates redundant links on divided highways. For instance, a one-way divided highway link is shown as a two-way divided highway link in Cube. Defining correct directionality of divided highway links is a crucial step to trip assignment. Eventually, the redundant links (link direction not consistent with road direction) on divided highways were removed manually in Cube.

3.2.2.6 Node with Gaps

Figure 3.7 is an example illustrating the connectivity problem between two nodes. In this example, nodes 423 and 424 appear to be connected, but in fact there is a small gap between them. The disconnected nodes would result in disconnected links, which are link 422-423 and link 424-425. The disconnected links would lead to inaccurate trip assignment results. To fix this problem, the *Grouping Limit* of 3.5 feet was specified in Cube when creating network from an ArcGIS file. The *Grouping Limit* function works in a way to connect nodes within the distance specified. In this study, several *Grouping Limit* values were tested and 3.5 feet works the best to create a well connected network.

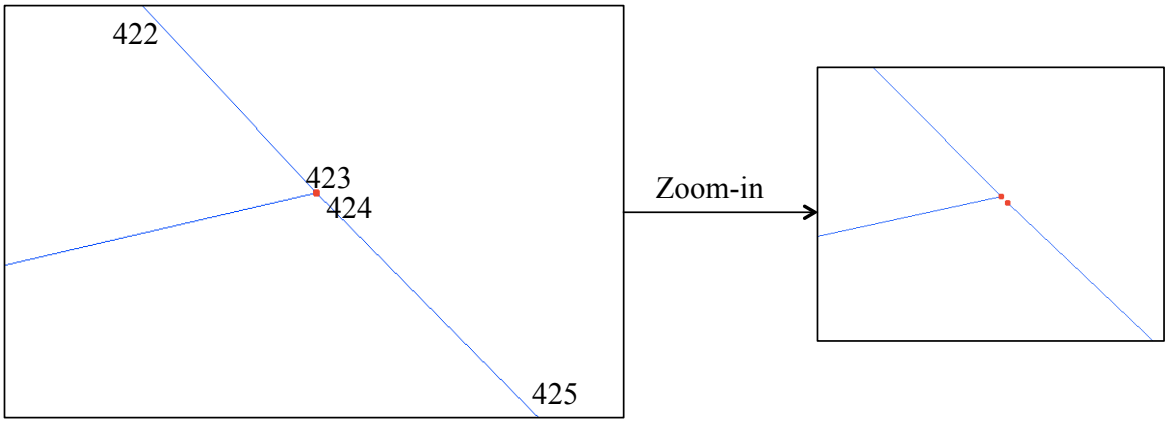

Figure 3.7 Example of disconnected nodes

3.2.3 Link Attributes

3.2.3.1 Functional Classification

The functional classification defines the character of service a roadway segment provides. It is an important variable to determine other link attributes, including free flow speed and link capacity.

The original definition of the functional classification categories (also functional class in short) was provided by the SHA in the functional class network. The SHA only defines seven links classes. To better represent the network attribute, functional class is also defined for ramp links and centroid connectors. Table 3.4 presents the functional class defined for the network system. Figure 3.8 shows the distribution of link functional class on the network.

Table 3.4 Definition of Functional Class

Functional Class	Definition
1	Interstate
2	Principal Arterial – Other Freeways and Expressways
3	Principal Arterial – Other
4	Minor Arterial
5	Major Collector
6	Minor Collector
7	Local
8	High Speed Ramp
9	Medium Speed Ramp
10	Low Speed Ramp
11	Centroid Connector

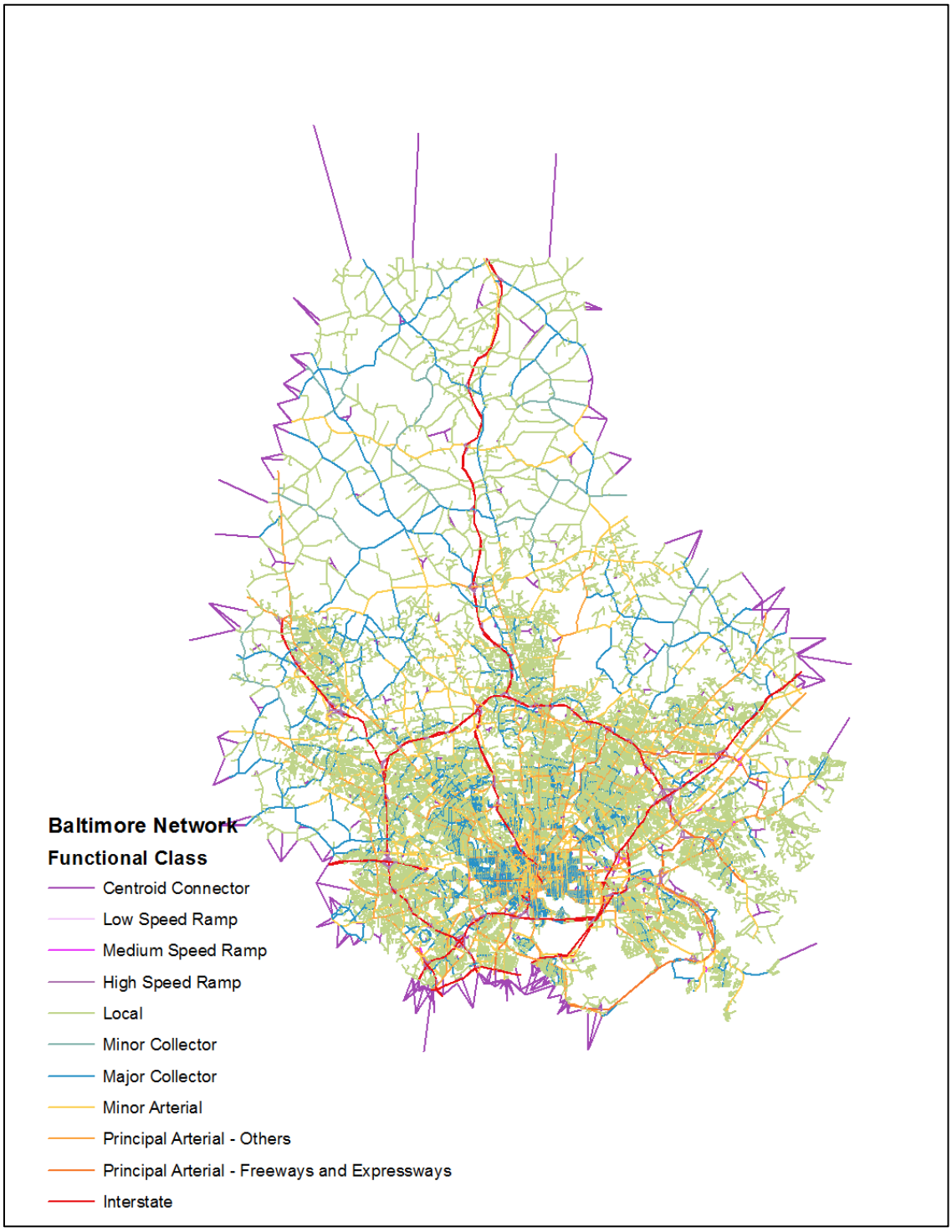

Figure 3.8 Link attribute - functional class in the Baltimore area

3.2.3.2 Number of Lanes

The number of lanes is defined for each link segment on the network, using the lane/speed limit network provided by the SHA. Figure 3.9 shows the distribution of number of lanes on the network. The number of lanes will be used to calculate link capacity on multi-lane highways.

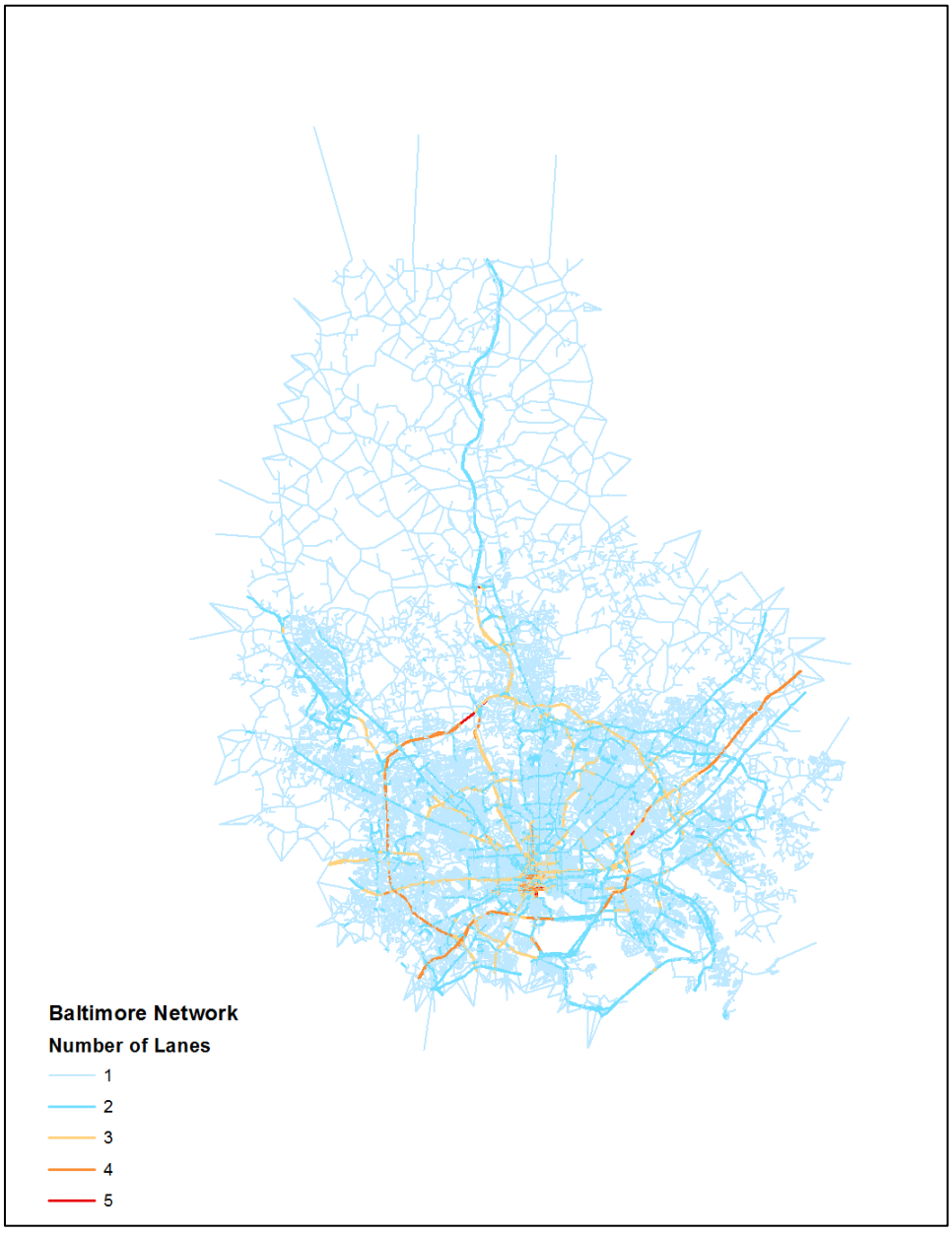

Figure 3.9 Link attribute - number of lanes in the Baltimore area

3.2.3.3 Free Flow Speed

The link speed is an important network attribute. In trip assignment, link speed is used to compute link travel time and volume. The Federal Highway Administration defines free flow speed as the "mean speed of passenger cars measured during low to moderate flows, usually up to 1300 passenger cars per hour per lane". In this study, the free flow speed is determined through a look-up table provided by the BMC *(15)*. The area type and functional class given in the BMC look-up table are not consistent with their definitions in the current study. To match the definitions of link attributes on the current network, the BMC table is modified accordingly as presented in Table 3.5.

Table 3.5 Look-up Table of Free Flow Speed Modified from the BMC Definition

Area Type / Functional Class	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Interstate SPDP*=>65	63	63	62	59	58	58	58	58	58
Interstate SPDP=60	58	58	56	56	55	55	54	54	54
Interstate SPDP=55	54	54	54	52	51	50	46	45	42
Interstate SPDP<55	52	52	48	46	46	46	42	41	38
Freeway SPDP=>65	62	61	60	58	57	56	56	54	54
Freeway SPDP=60	57	56	55	55	53	53	52	48	44
Freeway SPDP=55	53	53	53	51	50	48	47	45	43
Freeway SPDP<55	48	46	46	46	43	39	32	27	25
Principal Arterial	37	37	36	33	28	26	23	22	19
Minor Arterial	34	32	28	26	26	23	22	19	18
Collector	31	28	22	20	20	19	17	15	15
Interstate High Speed Ramp	46	46	46	46	46	46	44	44	44

Interstate Medium Speed Ramp	28	28	28	28	28	28	26	26	26
Interstate Low Speed Ramp	18	18	18	18	18	18	16	16	16
Freeway Medium Speed Ramp	22	22	22	22	22	22	22	22	22
Freeway Low Speed Ramp	14	14	14	14	14	14	14	14	14
Centroid Connector	30	25	25	25	20	20	20	15	15

* SPDP is posted speed

3.2.3.4 Link Capacity

The Highway Capacity Manual defines capacity as "the maximum sustained 15 minutes flow rate, expressed in passenger cars per hour per lane, that can be accommodated by a uniform freeway segment under prevailing traffic and roadway conditions in one direction of flow." *(16)* The Highway Capacity Manual discusses a broad range of factors that affect capacity, including area type, lane and shoulder widths, transit stops, truck movements, median treatments, intersection types, signal timing, etc. The Highway Capacity Manual also recommends defining capacity under Level of Service E, which is when traffic flow becomes irregular and speed varies rapidly.

Following the recommendations in the Highway Capacity Manual, the BMC defined the link capacity (vehicles per hour per lane) as a function of area type, functional class and number of lanes *(15)*. The BMC look-up table is modified to be consistent with the definitions of link attributes on the current network, as presented in Table 3.6.

Table 3.6 Look-up Table of Link Capacity Modified from the BMC Definition

Area Type / Functional Class	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Interstate 4-6 lanes	2400	2350	2350	2350	2300	2300	2250	2250	2200
Interstate 2-3 lanes	2200	2150	2150	2150	2100	2100	2050	2050	2000

Interstate 1 lane	1800	1750	1750	1750	1700	1700	1700	1600	1600
Principal Arterial multilane	2100	2050	2000	1950	1950	1900	1900	1900	1900
Principal Arterial 1-2 lanes	1500	1450	1400	1350	1350	1300	1300	1300	1300
Principal Arterial multilane	2100	2050	2000	1950	1950	1900	1900	1900	1900
Principal Arterial 1-2 lanes	1500	1450	1400	1350	1350	1300	1300	1300	1300
Principal Arterial Other multilane	1150	1150	1150	1100	1100	1050	1050	1000	1000
Principal Arterial Other 1-2 lanes	1050	1050	1050	1000	1000	950	950	900	900
Minor Arterial multilane	900	900	900	850	850	800	800	750	750
Minor Arterial 1-2 lanes	800	800	800	800	750	750	750	750	750
Major Collector & Minor Collector	800	800	800	800	750	750	750	700	700
Local	800	800	800	800	750	750	750	700	700
Interstate High Speed Ramp	2200	2150	2100	2050	2050	2000	2000	2000	2000
Freeway Medium Speed Ramp	2050	2000	2000	2000	1950	1950	1950	1900	1900
Freeway Low Speed Ramp	1900	1850	1850	1850	1800	1800	1800	1750	1750
Centroid Connector	3000	3000	3000	3000	3000	3000	3000	3000	3000

3.2.3.5 Toll Rate

In trip assignment, it takes road pricing into account when computing link travel time. In the study area, there are three toll roads in the year of 2007 (model's base year): (1) the Fort McHenry Tunnel is tolled in westbound and eastbound directions, each at a flat-rate of $4.00; (2) the Harbor Tunnel is tolled in westbound and eastbound directions, each at a flat-rate of $4.00; and (3) the Key Bridge is tolled in westbound and eastbound directions, each at a flat rate of $4.00.

3.3 Validation Data

Validation is an important component of travel demand models. The reproduction of observed traffic conditions by the modeling results is a critical validation criterion. The validation data were provided by the SHA on an hourly basis. The data represent observed traffic counts collected by the SHA at a number of count stations during certain days in the year of 2007. In this study, the count data used to validate the modeling results is defined as average daily PM counts, which were collected from the PM time periods (4:00 PM – 7:00 PM) in 2007.

According to the location description provided by the SHA, the count data were manually matched with links on the current network. In total, there are 210 count stations identified on the network, 42 of which are within the Baltimore City. Table 3.7 shows the number of count stations for each function class. Figure 3.10 shows the distribution of the count stations in the study area.

Table 3.7 Traffic Count Stations

Functional Class	Number of Stations
Interstate	25
Principal Arterial – Other Freeways and Expressways	16
Principal Arterial – Other	114
Minor Arterial	38
Minor Collector	12
Local	3

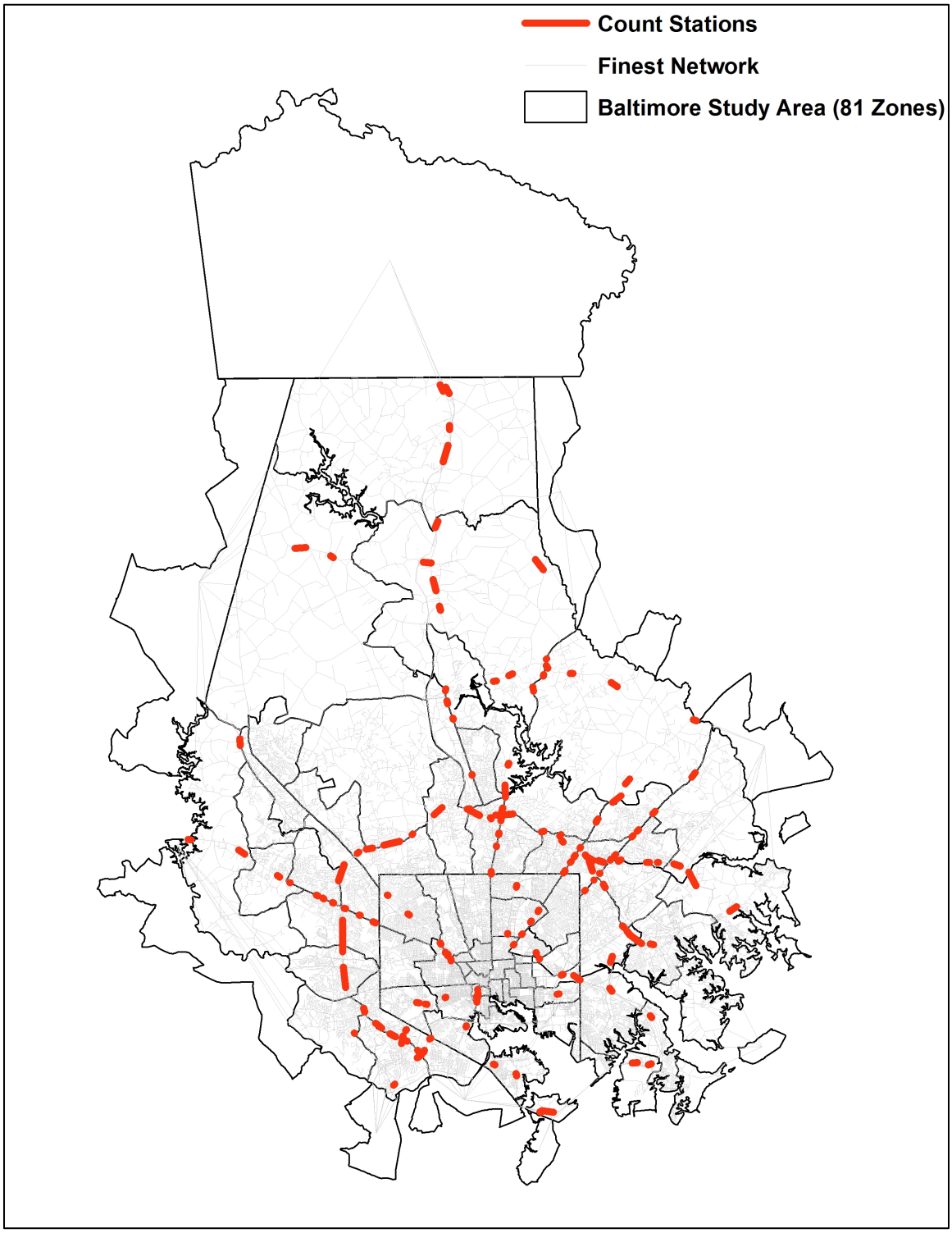

Figure 3.10 Traffic count stations in the Baltimore area

4 METHODOLOGY

In this chapter, the research methodology is presented. First, it develops a comprehensive model to define the network resolution. After that, it prepares the study area, the zone system and the finest network system in Section 4.2. Section 4.3 describes how to assign a trip table to the network and evaluation criteria of assignment results. Section 4.4 presents algorithms developed to identify irrelevant links on a network for a given zone system.

4.1 A Comprehensive Model to Define Network Resolution

Briefly, this model is an iterative process to remove irrelevant links starting from the finest network. Figure 4.1 shows the complete procedure. Each component of the model will be discussed in greater detail in the following sections.

First, this model performs the trip assignment (see Section 4.3) on the finest network. Then, it identifies links on the finest network that have not been assigned any volume. These links are irrelevant for the given zone system and should be removed (see Section 4.2.4). The assignment results are validated against observed traffic count data. Performance measures are calculated to evaluate if the modeled traffic volumes are close enough to the traffic conditions in reality (see Section 4.3.6).

After that, if the validation result is satisfactory, the program is terminated and the developed network is exported. If not, this program continues to identify irrelevant links (see Section 4.4) on the current network and removes them from subsequent iteration(s). The objective of the validation analysis is to (1) minimize the *%RMSE*, |*%DIFF*|, |*AE*|

and |*DSD*| values; and to (2) maximize the R^2 value. It is assumed that the level of network detail is consistent with the zone system when the objective is achieved.

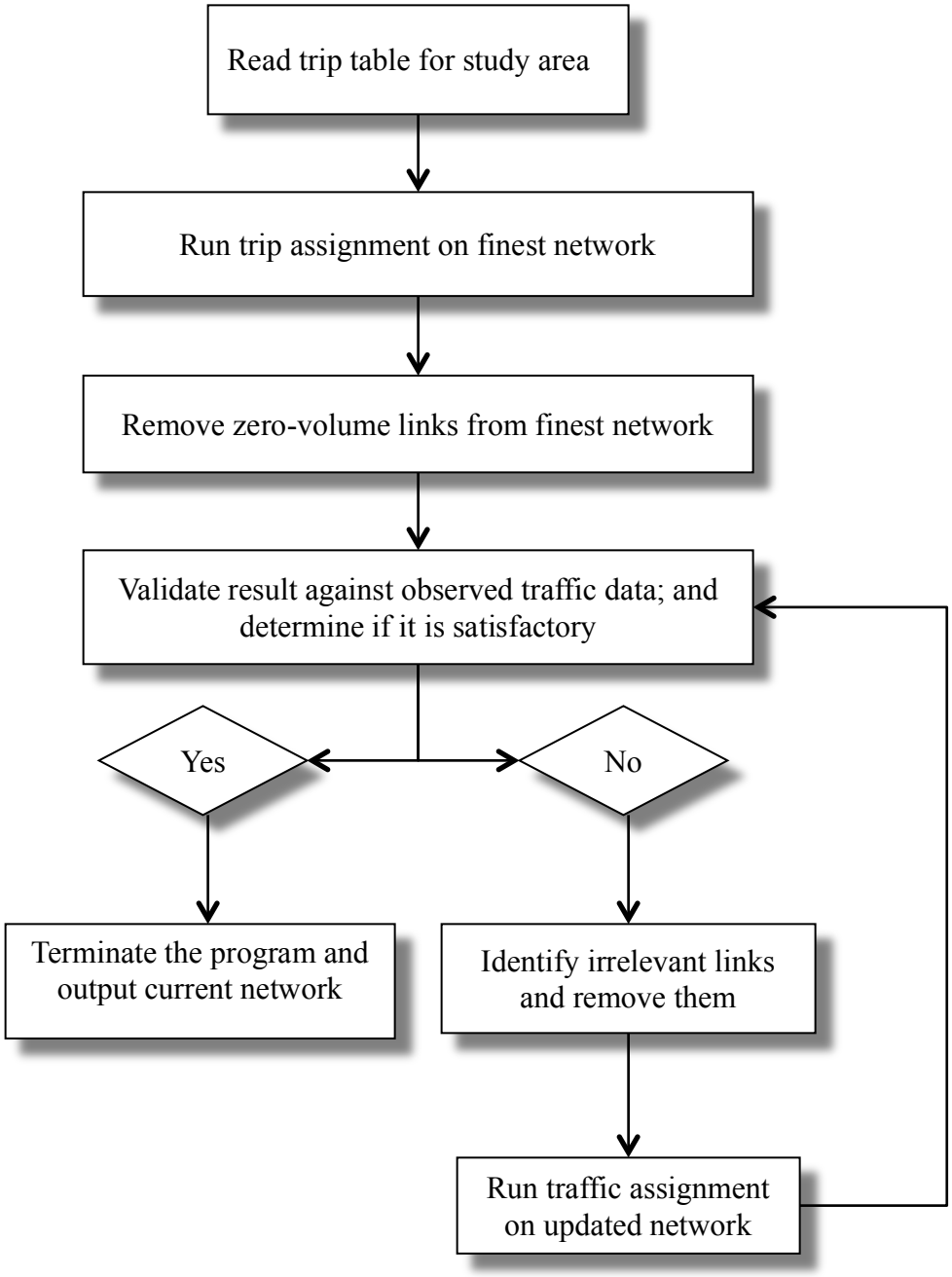

Figure 4.1 A comprehensive model to define network resolution

4.2 Study Area

The trip table, which represents the travel demand in the study area, is obtained after the trip generation, trip distribution and mode choice steps in a travel demand model. Originally, there are 1,674 TAZs delineated in the MSTM. In this dissertation, the study area has been defined within the Baltimore area (Baltimore County and Baltimore City). In order to acquire the trip interchanges in the study area, the Subarea Analysis is applied.

4.2.1 Subarea Analysis

The subarea analysis was conducted in Cube using the MSTM outputs of the person travel and truck travel models (See Section 3.1.1). It utilizes the original MSTM network and its zone system to extract the trip table for the study area. The subarea analysis is performed as follows.

First, the boundary of the Baltimore area is prepared in ArcGIS. In Cube's Network Program, the original MSTM network is imported as the Highway Layer; the map of the Baltimore area boundary is imported as the Boundary Layer, and both layers must be made visible, as presented in Figure 4.2. The blue links indicate the MSTM network links, and the highlighted polygon indicates the boundary of the Baltimore area.

Second, a subarea network is created using the *Sub-Area Extraction* tool in Cube. It renumbers the MSTM zones and nodes by consecutive numbers, as shown in Figure 4.3. On the subarea network, the zone range within study area is 1 – 298, the external zone range is 299 – 362, and the node range is 400 – 4408. Figure 4.4 shows the MSTM network representation of the Baltimore area.

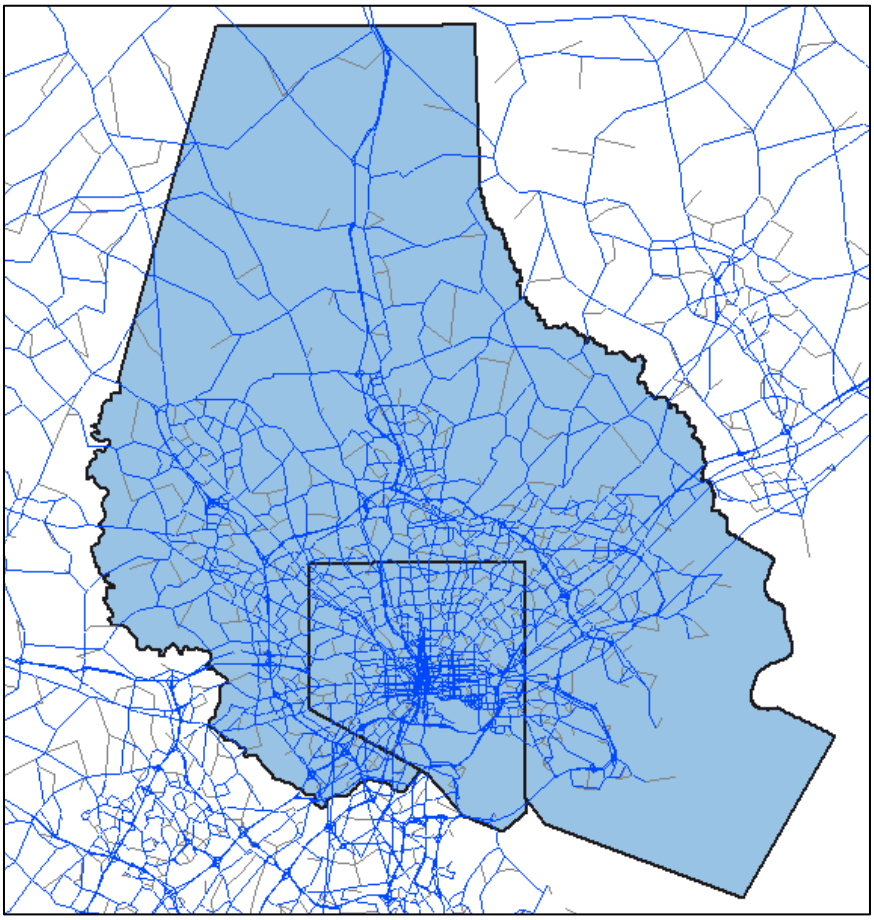

Figure 4.2 MSTM network and study area boundary displayed in Cube

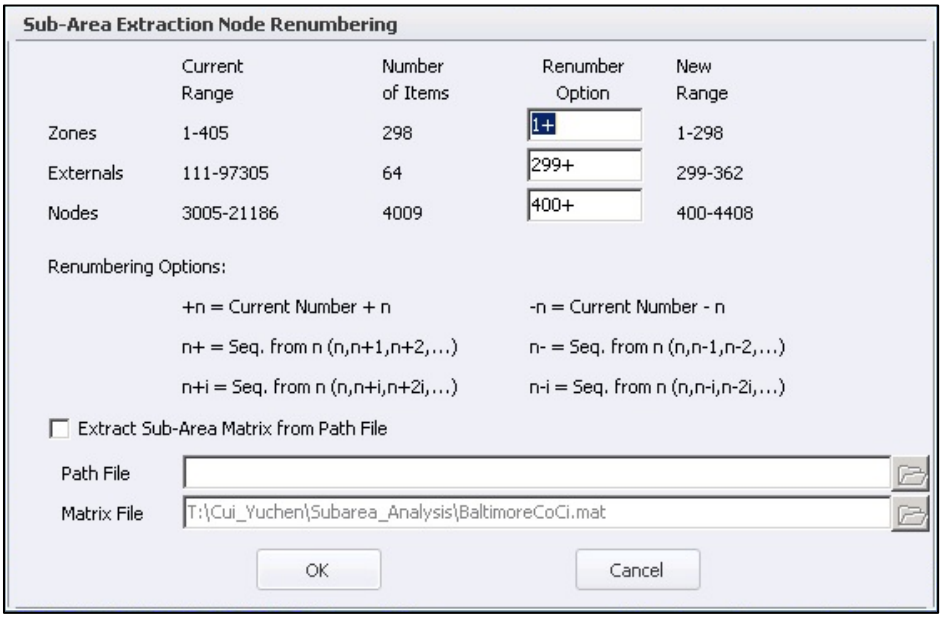

Figure 4.3 Subarea network extraction renumbering in Cube

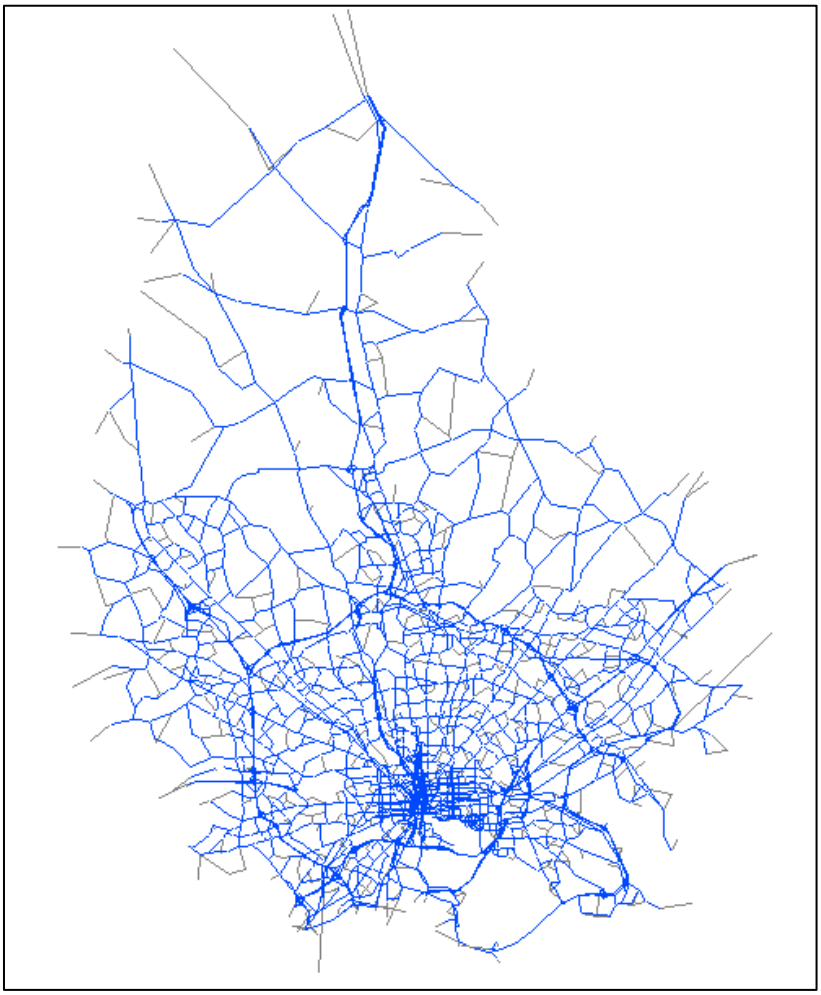

Figure 4.4 MSTM subarea network of the study area

The MSTM considers highway travel (automobile and truck) and transit trips (light rail, Amtrak, etc.) for four time-of-day periods. In this study, highway travel during AM (6:30 AM – 9:30 AM) and PM (3:30 PM – 6:30 PM) periods is considered. After extracting the subarea network, the trip assignment is performed, assigning trip tables on the subarea network to obtain the subarea trip tables for morning and afternoon peak hours, respectively.

4.2.2 Post-processing of Subarea Analysis

The subarea trip tables include OD interchanges for 362 zones, containing 20 vehicles classes: drive alone for five income groups, shared ride with two occupants for five income groups, shared ride with 3 occupants for five income groups, long-distance autos, commercial vehicles, short distance single-unit trucks, short distance multi-unit trucks, and long-distance trucks. Considering that the following analyses will evaluate the spatial consistency of the network and zone systems based on the overall trip pattern on the network, only the total amount of trips assigned on the network will be relevant. Eventually, the 20 trip tables (of 20 vehicle classes) were summed up to one trip table in Cube.

Furthermore, several model runs have been tested on zone systems differing in zone size. The testing results showed that having a coarser zone system could substantially increase computational efficiency of the model. Therefore, in the study area, the trip interchanges on the MSTM-defined zone system (362 TAZs) were aggregated to a coarser zone system (81 TAZs) by a Cube script. The aggregated zone system includes eight external zones and 74 internal zones.

As a result, the trip table used in the following analyses contains 81 TAZs and it accounts for the total number of vehicles traveling in and through the study area. Figure 4.5 presents the MSTM-defined zone system and the aggregated zone system. The external zones are highlighted in red, and they represent the locations where through trips (with at least one trip end within the Baltimore area) enter the internal zones.

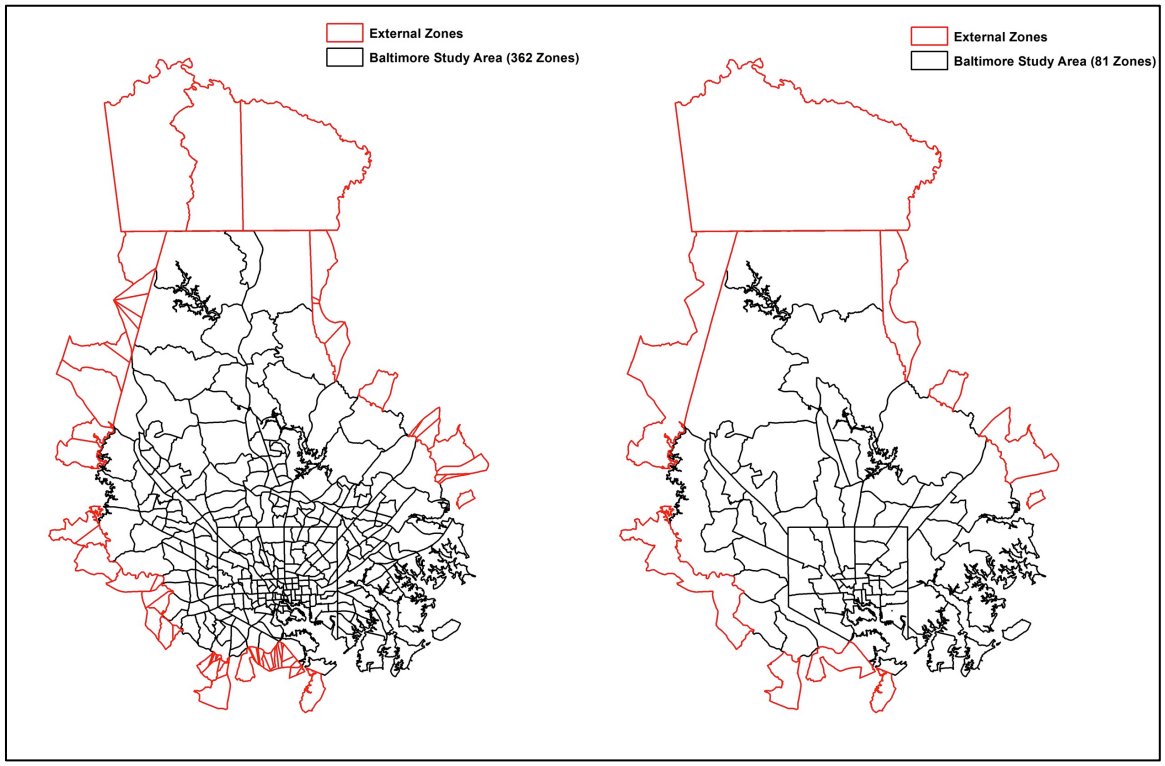

Figure 4.5 Zone system defined for the study area

4.2.3 Finest Highway Network

Given the defined zone system, the quest for an appropriate level of network resolution starts with the finest network system. This study only considers vehicle trips and highway network, excluding transit trips and transit network from the model development. In a travel demand model, transit trip assignment is usually performed separately from highway trip assignment, and transit trips are often assigned on fixed transit routes. That is to say, the assignment results of transit trips do not have any impact on the assignment results of highway trips. Therefore, only vehicle trips on highway network are represented in this study.

The finest network system was developed in ArcGIS and finalized in Cube based on the Centerline network and several road networks provided by the SHA. The data sources and definitions of network attributes have been described in Chapter 3.

The zone centroids and centroid connectors were created in Cube. The centroids are defined at the center (gravity point) of each zone. The centroid connectors are created for each zone, following several rules:

- The maximum number of centroid connectors is 4.
- The maximum distance for connectors is 4 miles.
- Only connect to minor collectors and local roads.
- Spread connections to the closest nodes in as many directions as possible.

This approach guarantees that there are sufficient centroid connectors created for each zone that can transport traffic flows in different directions on the network. The centroid connectors are made connect with lower class roads so that travellers can fulfill their trips by taking lower class roads first before getting onto higher class roads, then getting off from higher class roads to lower class roads before arriving at their destinations. This approach makes sure that trips will be assigned through centroid connectors onto the network in a way that is as close to real travel patterns as possible. Figure 4.6 shows the finest highway network and its centroid connectors.

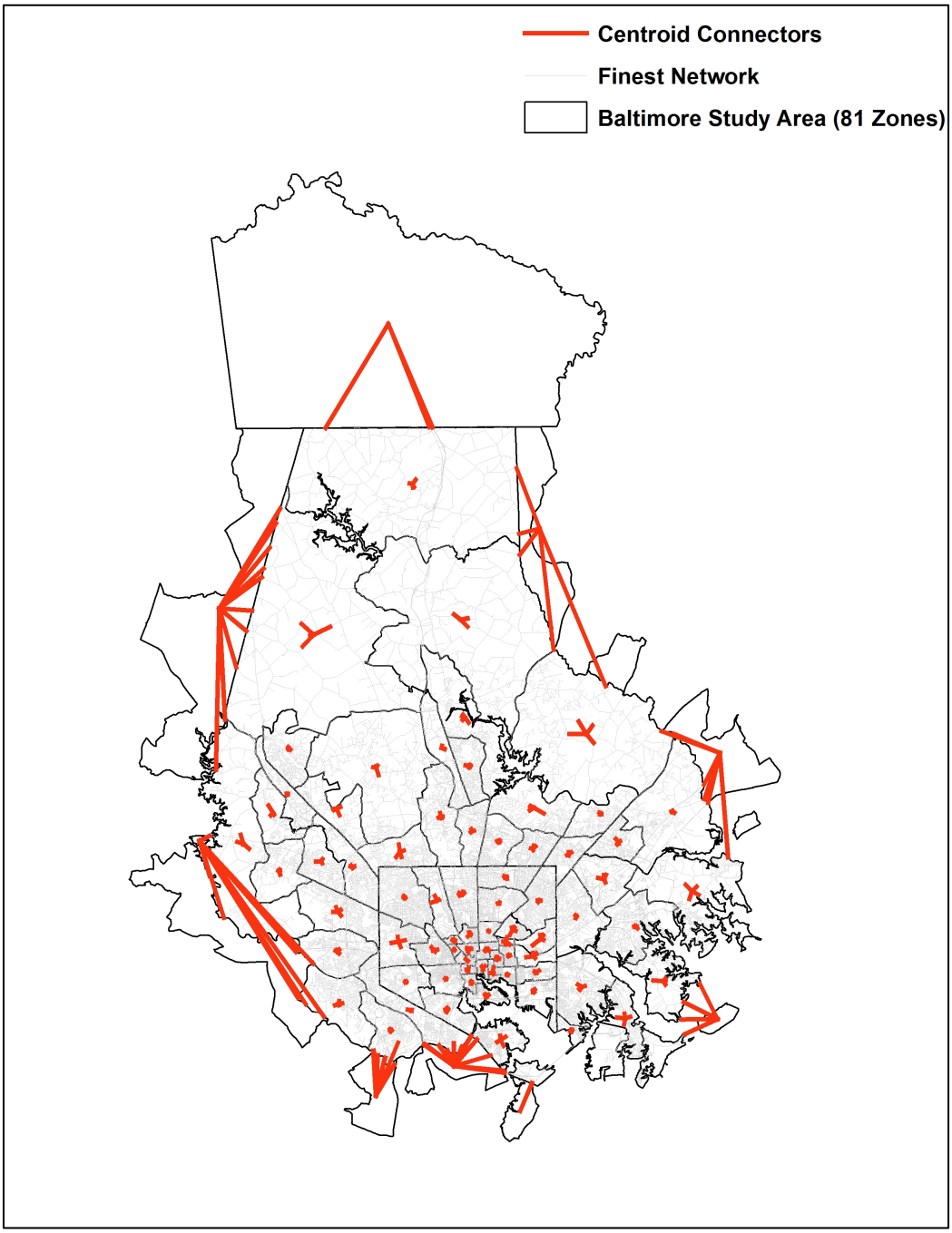

Figure 4.6 Finest highway network with centroid connectors generated from zone centroids

4.2.4 Modification of Finest Network

An initial trip assignment was performed on the finest network to test if it is routable. Visual inspection was conducted to evaluate the assignment results, including (1) all

interstate highway links should have volumes; (2) for those links on interstate highway, freeway, expressway, and major arterial that do not have volumes, check their connection with adjacent links; and (3) ramps should be properly assigned.

It is expected that the finest network is over-detailed compared to the highly aggregated zone system; therefore it might result in a portion of links that will not be used by any OD pairs. This group of zero-volume links is an indication that they are over detailed for the coarse zone system. Before conducting further analyses, these unused links should be removed from the finest network to accelerate the development of a more consistent zone-network system. Meanwhile, a network with fewer links can substantially save the model's computation time.

4.3 Trip Assignment

The quantitative analysis of transport movement over a physical network is performed with the aid of a network model, which presents the transport network and computes the traffic flows on the network links. Such models are descriptive of travellers' behavioral patterns and are usually referred to as network equilibrium models *(17)*. Various algorithms have been developed to solve network models by characterizing a way in which road users travel from origins to destinations on the corresponding network. This section will discuss trip assignment method, model convergence criteria, contributing factors to model's computation time, and evaluation method of assignment results.

4.3.1 Trip Assignment Method – All-or-nothing Assignment

The all-or-nothing assignment is the simplest assignment method. It assumes that all users experience no congestion and perceive the road attributes (e.g. travel cost) in the

same way, which means all users choose the same shortest path (i.e. travel time or travel distance) between OD pairs.

The all-or-nothing assignment method does not represent what road users would actually experience in reality. However, it is a basic assignment method used for other assignment techniques to reach a desirable assignment result.

4.3.2 Iterative Trip Matrix Loading Method – Method of Successive Averages

The behavioral assumption of network equilibrium models is the Wardrop's user-equilibrium *(18)*, which states "the journey times in all routes actually used are equal and less than those which would be experienced by a single vehicle on any unused route." When a network reaches a user-optimized equilibrium, no user may lower his/her generalized cost (sum of monetary and non-monetary costs of a trip) by changing to an alternative route. When congestion (delay) occurs on the network, the solution to the network equilibrium problem becomes a nonlinear cost-network equilibrium problem. Many assignment methods have been developed to approximate to the equilibrium conditions as described in the Wardrop's user-equilibrium principle. Discussion of the problem's nonlinear solutions is beyond the scope of this dissertation, and is described elsewhere *(19)*.

In travel demand models, some trip assignment methods have been developed to load trip tables onto the network in several iterations, taking into account the relationship of users' travel cost (time) and link flow. These methods compute link travel speed or travel time for the current iteration and use the results in the next iteration, until the assignment results achieve a predefined convergence level. The method of successive

averages (MSA) remains by far the most widely used iterative averaging method, which produces a heuristic solution of network equilibrium problem. Generally, the MSA calculates network flows for current iteration by taking a linear combination of network flow from a previous iteration and an all-or-nothing assignment result for the current iteration *(19, 20)*. In this dissertation, the MSA loads the trip table obtained in Section 4.1.2 at zone centroids and then assigns it onto the highway network through centroid connectors by five steps.

Step 1: Initialize all flows to zero, and set the current iteration to zero.

Step 2: Update link cost at the current iteration, and update n = n + 1.

$$Cost_a = T_0 + \left(\frac{TOLL_a}{VoT}\right) + 0.25 \times DISTANCE_a$$

$$T_0 = 60 \times (DISTANCE_a / FFSPD_a)$$

where

$Cost_a$ is the cost on link a

$TOLL_a$ is the toll rate on link a

$DISTANCE_a$ is the link length of link a

VoT is a traveler's value of time; it is 8.4 cents/minute

$FFSPD_a$ is the free flow speed on link a

Step 3:Perform an all-or-nothing assignment and calculate auxiliary flows.

Step 4: Calculate network flows at the current iteration as

$$V_a^k = (1 - \emptyset)V_a^{k-1} + \emptyset F_a$$

where

k is the current iteration

V_a^k is the current flow on link a

$\emptyset$ is $1/k$

F_a is the auxiliary flow (calculated by all-or-nothing) on link a

Step 5: At the end of the current iteration, model convergence criteria are used to evaluate the solution's approximation to Wardrop's user-equilibrium. If the criteria are met, stop; otherwise proceed to Step 2.

4.3.3 Path-tracing between Origins and Destinations

The trip assignment is a two-step procedure. First, paths are built from each origin zone to every destination zone that it connects to. The second step is to load each trip interchange of the trip table to the path(s) between the OD pair it represents. When a travel path is built using a certain link, the traffic volume carried by this path will be loaded onto this link. After the entire trip table is assigned on the network, the assignment result on a link represents the accumulated volumes from trip interchange(s) that have built path(s) using this link.

In order to identify links that are irrelevant to represent trip interchanges in the study area, it is important to know how many trips that each individual link carries for each OD pair. Figure 4.7 presents the path-tracing model from origin *i* to destination *j* for all selected OD pairs.

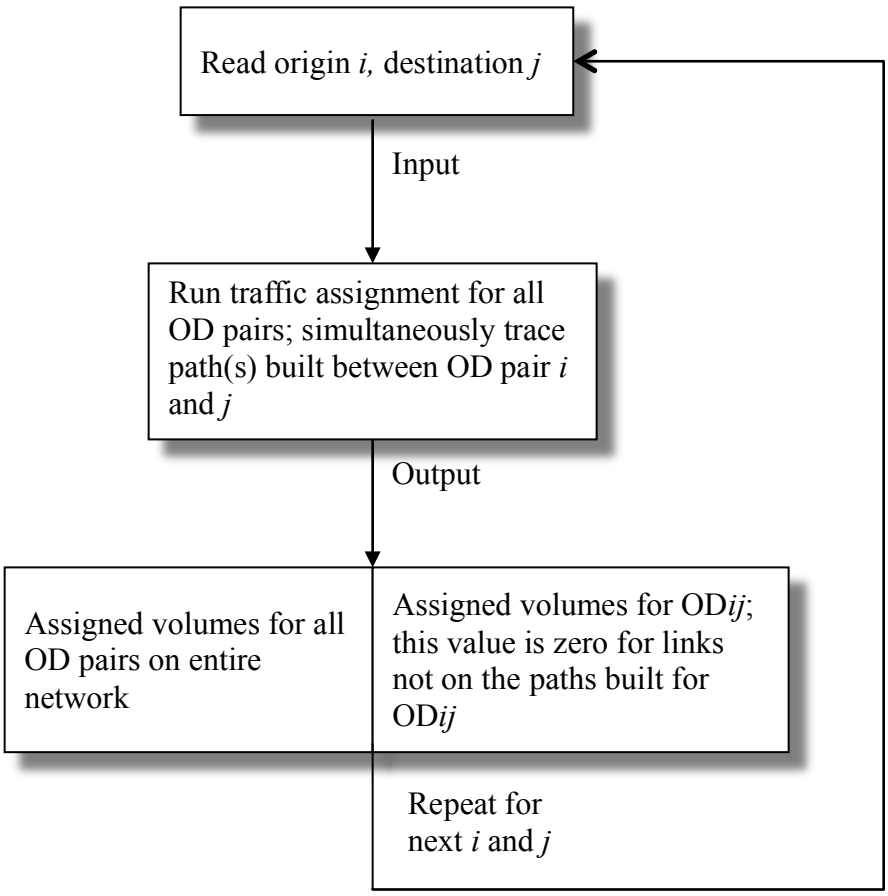

Figure 4.7 Path-tracing method applied in trip assignment

The procedure presented in Figure 4.7 was developed in Cube utilizing its SELECTLINK function. When implementing the path-tracing model, it starts from the OD pair with highest trip interchange and proceeds to the next OD pair in descending order. Sorting all OD pairs from highest trip interchange can help select the most important OD pairs, which carry a large enough proportion of the travel demand in the study area to identify irrelevant links. Given limited computing resources, it would be more efficient to analyze selected OD pairs than running the model for all OD pairs.

To sort OD pairs from highest trip interchange, a Cube script was developed to read in the trip table and write out its trip interchanges as individual records. A Java code

was developed to index interzonal trip interchange records, and sort them in descending order. Figure 4.8 presents the trip table transformation and sorting procedure.

Origin \ Destination	1	2	3	4	5
1	20559	1139	1187	332	331
2	1319	1892	1419	205	224
3	1362	1426	10664	1127	1671
4	273	154	891	6657	1259
5	310	186	1465	1352	6982

a. Trip Matrix in Cube

Order	Index
0	5775
1	6247
2	5774
3	2663
4	3980
5	5099

d. Sorted Result

Origin	Destination	Trip Interchange
1	1	20559
1	2	1139
1	3	1187
1	4	332
1	5	331

b. Individual Records for Trip Interchanges

Index	Origin	Destination	Trip Interchange
0	1	2	1139
1	1	3	1187
2	1	4	332
3	1	5	331
4	1	6	198
5	1	7	1057

c. Indexed Individual Trip Interchanges

Figure 4.8 Example of trip matrix transformation and sorting procedure

After the path-tracing model is completed, its results are further processed in Cube. For each OD pair of the model, the Cube script writes out a separate assignment result file, which only contains links on the path(s) built for this OD pair.

4.3.4 Model Convergence Criteria

In order to compare the assignment solutions computed by different assignment techniques, several criteria have been developed by travel demand modelers. The most important criterion is the relative gap, which is often used to measure how close an approximation is to Wardrop's user-equilibrium *(21)*. The relative gap is calculated as follows.

$$REL.GAP = \frac{\sum_{i=1}^{N}(VE_{k-1} \times COSTE_{k-1}) - \sum_{i=1}^{N}(VA_k \times COSTE_{k-1})}{\sum_{i=1}^{N}(VE_{k-1} \times COSTE_{k-1})}$$

where

k is the current iteration

N is the number of links

VE_k is the link equilibrium volume for iteration k

$COSTE_k$ is the link travel cost based on the equilibrium volume VE_k

VA_k is the link volume from an all-or-nothing assignment to the minimum cost paths based on $COSTE_{k-1}$

The relative gap measures the excess travel cost over the entire network of the current solution compared to the travel cost of an equilibrium solution. The relative gap would be zero when the current solution reaches Wardrop's user-equilibrium condition. A perfect equilibrium solution, or a zero-value relative gap, is rarely achieved due to the non-linearity of the equilibrium problem *(22)*. The level of convergence achieved by an assignment technique is highly dependent on the size of the network, number of zones, level of congestion, and number of vehicle classes. It has been recommended that a relative gap of 0.0001 should be used to assure that the assignment result is an equilibrium solution *(22)*.

Sometimes, results of several other criteria are reported to help assess if an assignment result reaches the desirable level of convergence. These criteria include the Average Absolute Volume Difference (AAD), Gap and Root Mean Squared Error of the Differences (RMSE).

The AAD calculates how successive iterations differ in their average modeled volumes.

$$AAD = |\bar{V}_k - \bar{V}_{k-1}|$$

where

$\bar{V}_k$ is the average of modeled traffic volume at the *k*th iteration

$\bar{V}_{k-1}$ is the average of modeled traffic volume at the *k-1*th iteration

The Gap calculates relative difference between two successive iterations in their modeled volumes.

$$GAP = \frac{\left|\sum_{i=1}^{N}(VE_k \times COSTE_k) - \sum_{i=1}^{N}(VE_{k-1} \times COSTE_{k-1})\right|}{\sum_{i=1}^{N}(VE_{k-1} \times COSTE_{k-1})}$$

where

k is the current iteration

N is the number of links

VE_k is the link equilibrium weighted volume for iteration *k*

$COSTE_k$ is the link travel cost based on the equilibrium volume VE_k

VA_k is the link volume from an all-or-nothing assignment to the minimum cost paths based on $COSTE_{k-1}$

The RMSE calculates the root mean squared error of the difference in modeled volumes between two successive iterations.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(VE - VE')^2}{N}}$$

where

N is the number of links

VE is the link equilibrium weighted volume for the current iteration

VE' is the link equilibrium weighted volume for the previous iteration

In this dissertation, the relative gap will be used as the primary convergence criterion to determine the convergence level of an assignment result. Other convergence criteria will be reported when comparing two successive iterations of an assignment result.

4.3.5 Computational Efficiency

When computing a trip assignment solution, the computational efficiency depends on several factors. The number of links in the network system and number of zones define the problem complexity; the computing environment determines how fast the machine can calculate the problem; and the convergence level decides when the program can stop for a desirable solution.

When implementing this methodology, efforts were made to reduce the computation time to find the most appropriate network resolution. Specifically, the zone size has been reduced and a looser convergence criterion has been applied. However, for the development of a production model in the future, original zone system of a travel demand model should be in use, and a more strict convergence level should be applied.

4.3.6 Performance Measures

In order to measure how well the assignment results represent traffic conditions in reality, it is important to validate the results against observed traffic data. Several performance measures are identified and calculated to evaluate the validation results.

Correlation Coefficient

The correlation coefficient is applied to evaluate the extent to which the modeled volumes and actual traffic counts are correlated.

$$R = \frac{1}{N}\sum_{i=1}^{N}\frac{(V_e - \overline{V_e})(V_c - \overline{V_c})}{\sigma_e \sigma_c}$$

where

N is the number of links

$\overline{V_e}$ is the average of modeled traffic volumes

V_e is the modeled link volume

$\overline{V_c}$ is the average of observed traffic volumes

V_c is the observed link traffic volume

σ_e is the standard error of modeled volumes

$\sigma_c$is the standard error of observed traffic volumes

Root Mean Square Error (RMSE) and Percent Root Mean Square Error (%RMSE)

The RMSE and %RMSE measure the variance between modeled volumes and observed traffic counts.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(V_e - V_c)^2}{N}}$$

$$\%RMSE = \frac{100}{\overline{V_c}}\sqrt{\frac{\sum_{i=1}^{N}(V_e - V_c)^2}{N}}\ \%$$

where

N is the number of links

V_e is the modeled link volume

$\overline{V_c}$ is the average of observed traffic volumes

V_c is the observed link volume

Average Error (AE), Difference between Standard Deviations (DSD)

The AE and DSD evaluate the systematic error as a result of spatial aggregation *(4)*. These two performance measures evaluate the differences between modeled volumes and observed traffic counts that are caused by a change in the level of network detail. In this study, this difference is the result of forcing the same amount of trips to be assigned over a smaller network (after network reduction).

$$AE = \overline{V_e} - \overline{V_c}$$

$$DSD = SD_e - SD_c = \sqrt{\frac{\sum_{i=1}^{N}(V_e - \overline{V_e})^2}{N-1}} - \sqrt{\frac{\sum_{i=1}^{N}(V_c - \overline{V_c})^2}{N-1}}$$

where

N is the number of links

$\overline{V_e}$ is the average of modeled traffic volumes

V_e is the modeled link volume

$\overline{V_c}$ is the average of observed traffic volumes

V_c is the observed link volume

Percent Difference

The difference between total observed counts and total modeled traffic volumes is measured. The percent difference evaluates if the current network overestimate or underestimate the traffic conditions as a whole.

$$\%\ DIFF\ = \frac{\sum_{i=1}^{N} V_e - \sum_{i=1}^{N} V_c}{\sum_{i=1}^{N} V_c}$$

where

N is the number of links

V_e is the modeled link volume

V_c is the observed link volume

4.4 Identify Irrelevant Links

The essence of this methodology is to identify links that are not relevant to represent the trip interchanges in the given zone system, and remove these links from the network system. It starts with the finest network developed for the Baltimore area, and removes links iteratively.

As defined in this study, the irrelevant links, which predominately carry intrazonal trips, are not important to carry interzonal trips between OD pairs. For a given zone system, its irrelevant links are assumed not to carry a significant amount of trips for any OD pairs. However, inclusion of irrelevant links on the network would lead to an underestimation of traffic conditions on major routes. The criterion used for the identification of irrelevant links is defined as follows.

Let *N (V, L)* represent a network system, where *V* is the set of nodes and *L* is the set of links. Let *Z (O, D)* represent a zone system, where *O* is the set of origins and *D* is

the set of destinations, and $O \in V$, $D \in V$. Let v_i^o be the volume on link i originated from zone o; v_i^d be the volume on link i destined to zone d; and v_i^{od} be the volume on link i originated from zone o and destined to zone d. Let v_i denote the volume on link i, and

$v_i = \sum_{o \in O} v_i^o = \sum_{d \in D} v_i^d = \sum_{o \in O} \sum_{d \in D} v_i^{od}$, $i \in L$, $o \in O$, and $d \in D$,

which is a summation over assigned volumes on link i for all OD pairs.

Let $M = (m_o^d)$ be a trip matrix, which contains number of o rows and number of d columns. Let m_o^d represent the trip interchange for OD pair od.

Let $TR_i^{od}\left(v_i^{od}\right) = v_i^{od}/m_o^d$ represent the trip share of link i for an OD pair od, which is calculated by dividing link volume by trip interchange between OD pair od.

Let K^* denote a set of irrelevant links, where $K^* \in L$. The identification of K^* follows several steps.

Step 1: Calculate link trip share $TR_i = \max\left(TR_i^{od}\left(v_i^{od}\right)\right)$, which represents the maximum trip share that link i carries.

Step 2: Specify a^* or maximum number of K^* that can be identified on $N\ (V, L)$. Compute $a^* =$ number of K^* / number of L. The set of K^* is composed of links with smallest TR_i on $N\ (V, L)$.

Step 3: Identify TR^* denoting the cutoff point between irrelevant and relevant links. For any identified irrelevant link, if its link trip share $TR_k > TR^*$ $(\forall\ k \in K^*)$, adjust the maximum number of irrelevant links (a^*) as specified in step 2, such that $TR_k \leq TR^*$ $(\forall\ k \in K^*)$.

Step 4: Remove K^* from $N\ (V,\ L)$, run trip assignment on $N\ (V,\ L - K^*)$, and provide performance measures for $N\ (V,\ L - K^*)$.

An example is given below to illustrate how this method (steps 1, 2, and 3) identifies irrelevant links for a given zone system. Table 4.1 presents the trip table of a given three-zone system, where there are 20 vehicles traveling from origin 1 to destination 3. Figure 4.9 shows the path-tracing assignment result from origin 1 to destination 3.

Table 4.1 Example of Identification of Irrelevant Links – Trip Table

Destination / Origin	1	2	3	Total
1	10	15	20	45
2	25	5	30	60
3	5	20	10	35
Total	40	40	60	140

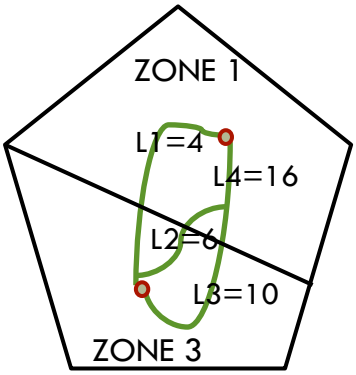

Figure 4.9 Example of identification of irrelevant links – trip assignment

The calculation of trip share from origin 1 to destination 3 is presented in Table 4.2. Same procedure is performed for the rest of OD pairs in the given zone system, as presented in Table 4.3.

Table 4.2 Example of Identification of Irrelevant Links – Trip Share Calculation

Link	Volume	Trip Share
1	4 vehicles	4/20 = 20%
2	6 vehicles	6/20 = 30%
3	10 vehicles	10/20 =50%
4	16 vehicles	16/20 = 80%

Table 4.3 Example of Identification of Irrelevant Links – Trip Share Results

O-D Pair / Link	1-2	1-3	2-3	2-1	3-2	3-1
1	15%	20%	18%	15%	17%	10%
2	20%	30%	7%	0	0	0
3	0	50%	87%	33%	19%	0
4	0	80%	25%	9%	0	15%
...	...	...	...	...	...	...
20	0	0	35%	25%	0%	58%

If an initial α^* is identified as 10% or 2 links on the current network containing 20 links, link 1 and link 2 are identified as irrelevant links because their link trip shares (TR_1 = 20% and TR_2 = 30%) are the smallest two out of the 20 links. If TR^*= 20% is applied to identify irrelevant links, it is found that TR_2 is greater than the TR^*. After adjusting α^* = 5%, eventually link 1 is identified as irrelevant link on the current network for the given zone system.

5 RESULTS AND ANALYSES

Results and analyses of the subarea analysis and the network-defining model are presented in this chapter.

5.1 Results of Subarea Analysis

The subarea analysis was performed for the AM peak period and PM peak period. As discussed in Section 4.2.2, 20 vehicle classes were combined into one to capture the entire traffic movement in the study area. As found in the results, total number of trips during the AM peak period is 860,789; and total number of trips during the PM peak period is 1,304,024. It is obvious that the study area experiences busier traffic during the PM peak period.

Therefore, trip table of the PM peak period is utilized to represent the travel demand of the given zone system. It can be expected that, with busier traffic movement in the study area, a change in the network resolution would result in an obvious change in the assignment results.

5.2 Adjustment of Computation Time

The proposed method to define the network resolution for a given zone system requires an iterative model-run of the trip assignment. Given the limitation of the current software used for trip assignment, it can only trace the path-building for one OD pair at a time. The 6,561 OD pairs (81 origins by 81 destinations) on the current zone system would need to run the trip assignment 6,561 times to get the complete path-building results for all OD pairs. Due to the limited computing resources when developing the methodology, several adjustments were made to reduce the model's computation time.

Some adjustments have already been addressed in previous sections. In the beginning of this study, the zone system was reduced from 362 TAZs to 81 TAZs (see Section 3.1.3). By using a coarser zone system, it works equally well to develop and implement the methodology, while the model runtime on the coarser zone system is 70% of that on the original zone system, all else being equal. After all, the methodology should be able to develop a network for any given zone system. In Section 4.2.2, 20 vehicle classes (in 20 trip tables) were aggregated into one vehicle class (in one trip table), representing the total travel demand.

To further accelerate the entire modeling procedure, several adjustments will be made, including removing irrelevant links from the initial finest network, loosening the convergence level, and analyzing a selected number of OD pairs for the identification of irrelevant links.

5.2.1 Convergence Level

Given the current computing environment (Windows Server 2008 R2, Intel(R) Xeon(R) CPU, 16 processors), the MSA algorithm applied in the trip assignment approximates to the user-equilibrium solution faster than other assignment algorithms. Table 5.1 presents the progression of model convergence, until the relative gap reaches a desirable level (relative gap = 0.0001) at the 318th iteration.

Table 5.1 MSA Convergence Progression Based on Finest Network

Iteration	Relative Gap	Root Mean Square Error (RMSE)	Gap	Average Absolute Volume Difference (AAD)
1	0	--	0	--
2	0.89488	2,411	0.47113	1,412
3	0.8213	1,089	0.49924	635

4	0.59275	590	0.31501	349
5	0.44471	409	0.21148	234
6	0.22642	336	0.07094	180
7	0.21465	238	0.06422	134
8	0.11071	197	0.02363	109
9	0.11429	169	0.0023	91
10	0.08739	148	0.00034	78
...	...	...	...	...
310	0.00022	3	0.00014	2
311	0.00013	3	2.49E-06	2
312	0.00023	3	0.00011	2
313	0.00027	3	4.41E-05	2
314	0.00081	3	0.00015	2
315	0.00041	3	8.17E-05	2
316	0.00037	3	5.41E-05	2
317	0.00010	3	1.70E-05	1
318	0.00004	3	2.71E-05	2

The predefined convergence level has a great impact on the model runtime. It is a trade-off between longer model runtime and closer approximation to the user-equilibrium solution as well as improved validation results. Table 5.2 compares the model runtime and validation statistics by applying different convergence criteria.

Table 5.2 Validation Results of Different Convergence Levels Applied to Finest Network

Relative Gap	**Runtime**	**Number of Iterations**	R^2	**%RMSE**	**%Diff**	**Ave. Error**	**Diff. of Std. Dev.**
0.0001	26 min 08’	318	0.865599	43.70%	-3.11%	-142.210	-417.515
0.0005	25 min 37’	296	0.865934	43.72%	-3.02%	-166.952	-396.120
0.001	21 min 36’	255	0.866004	43.70%	-3.00%	-138.203	-435.402
0.005	7 min 51’	93	0.867960	43.08%	-2.44%	-112.396	-419.362
0.01	4 min 29’	50	0.869527	42.59%	-1.97%	-91.039	-396.863
0.02	2 min 44’	31	0.872036	41.90%	-1.39%	-63.965	-363.425

On the current machine, if the convergence level is set at relative gap = 0.01, analyzing 1,000 OD pairs takes only 21% of the time by setting relative gap at 0.001. Obviously, the model runtime could be saved substantially by applying a looser convergence level.

5.2.2 Zero-volume Links on Finest Network

After the finest network was developed in Section 4.2.3, the MSA algorithm assigned the PM peak trip table on it.

Figure 5.1 presents the initial trip assignment results. The finest network with assigned volumes was scrutinized. Overall, the assignment results are consistent with expectation, and some specific observations can be made: (1) links on interstate highways are well connected with ramp links; (2) links with higher functional class have larger traffic volumes assigned to them; and (3) a number of local links do not have any assigned volumes. Generally, it is a good practice to examine zero-volume links to identify network errors, such as connectivity and directionality issues. Table 5.3 summarizes the zero-volume links by functional class on the finest network. The large portion of zero-volume links (62.85%) is due to the inconsistent spatial resolution between the detailed network and highly aggregated zone system. However, the higher class links with no volume assigned are worth a careful examination.

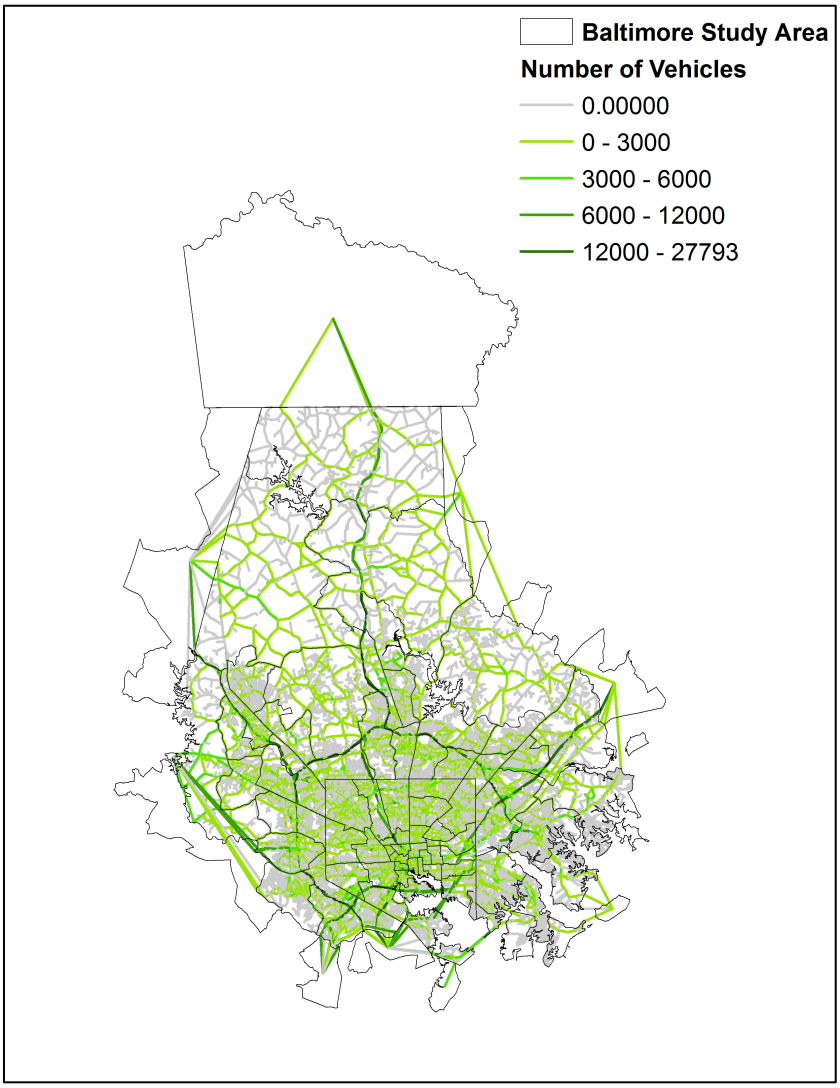

Figure 5.1 Trip assignment results on the finest network

Table 5.3 Links with Zero Volume in the PM Assignment on Finest Network

Functional Class	Number of Zero-volume Links	Total Number of Links	Percent of Total Zero-Volume Links
Interstate	35	908	3.85%
Principal Arterial – Other Freeways and Expressways	68	495	13.74%
Principal Arterial – Other	541	9,071	5.96%
Minor Arterial	3,440	15,118	22.75%
Major Collector	12,350	18,828	65.59%
Minor Collector	3,798	4592	82.71%
Local	64,501	84,739	76.12%
High Speed Ramp	154	802	19.20%
Medium Speed Ramp	36	109	33.03%
Low Speed Ramp	9	12	75.00%
Centroid Connector	150	690	21.74%
Total Zero-volume Links	85,082	135,364	62.85%

For the zero-volume interstate highway links, most of these links have been mistakenly identified as “interstate highway” in their network attribute due to the inconsistent network representations of different data sources within the Baltimore City (see Section 3.2.3.1), where most of these links are located. In a few cases, the zero-volume interstate highway links are dangling links due to the connectivity issues in the original Centerline network. It can be noticed that, some arterials are not used either and most of them are in the Baltimore City. The reason is that the grid network in the Baltimore City provides many alternatives to road users.

Visual inspections were conducted on entrance and exit ramps as well. Having checked the connection between ramps and adjacent links, it is found that the group of unused ramps is the result of the inconsistent network and zone systems. For instance, the zero-volume ramps at some interchanges are due to the low traffic volumes at these locations. The detailed network gives road users other alternatives so that they have bypassed the zero-volume ramps and taken others exits/entrances to lower their travel time. There are a number of collector and local links that do not receive any volume, which is also the consequence of having inconsistent network and zone systems.

Removing the zero-volume links is assumed to have no influence on the modeling results but could potentially reduce the model’s computation time. Therefore, these links are eliminated from the finest network and the rest of the links form the modified finest network.

5.2.3 Modified Finest Network

Figure 5.2 shows the assignment results after removing links that were not assigned any volume on the finest network. The model runtime was reduced from 4 min 29 sec for the finest network to 4 min 08 sec for the modified finest network. Their validation results are compared and presented in Table 5.4. The comparison of validation results reveals that the modified finest network has similar validation results as the original network. The slight differences in the validation statistics are due to the removal of eight zero-volume links where count stations are placed. To keep the validation analysis consistent throughout the study, only validation links that have modeled volumes are used in the analysis.

Table 5.4 Comparison of Validation Results of the Fines Network and Its Modification

Number of Zones	Number of Links	R^2	RMSE	% RMSE	% Diff.	Average Error	Difference of Std. Dev.
81	135,364	0.867022	1861.6380	42.82%	-2.70%	-120.59	-360.877
81	48,339	0.870226	1847.3933	42.59%	-2.21%	-101.83	-399.727

According to the percent difference in Table 5.4, it can be seen that the total modeled volumes on the count stations are lower than the observed traffic counts, indicating that the traffic condition is underestimated on the current network.

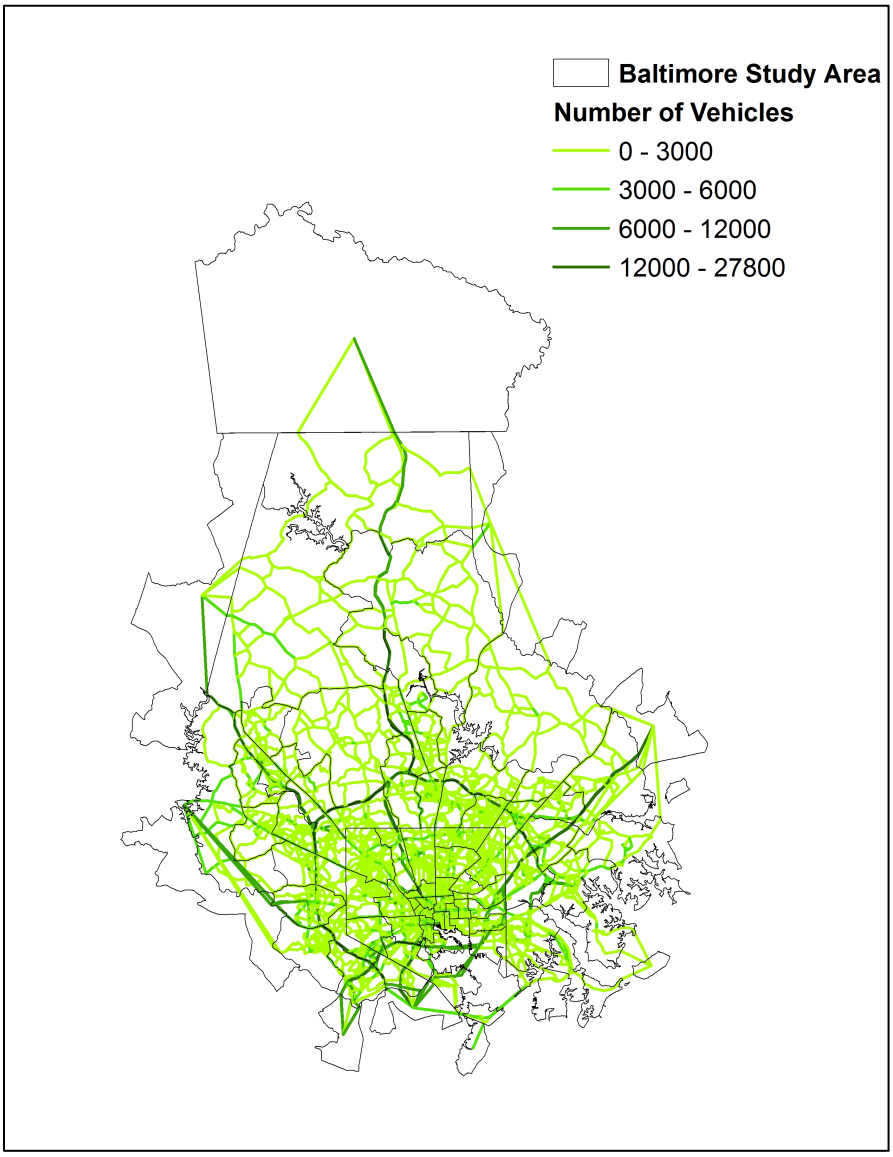

Figure 5.2 Trip assignment results on the modified finest network

5.3 Identify Irrelevant Links

As proposed in Section 4.4, it is essential to retrieve the path-building records for all OD pairs in the study area to identify the irrelevant links. Then, trip shares are calculated for each link and for every OD pair it belongs to. This method is implemented through several modeling components, most of which were developed in Java.

5.3.1 Sort Origin-Destination Trip Table

There are 6,561 OD pairs in the study area. If unlimited computing resources were available, analyzing all 6,561 OD pairs would be preferable. However, given the available computing resources, running the path-tracing model for 6,561 OD pairs would take 18 days. To make this process more efficient, the most important 1,000 OD pairs (intra-zonal OD pairs excluded) were chosen to identify the irrelevant links. The trip table is sorted in descending order in terms of their trip interchange values. Table 5.5 presents sorting results and Figure 5.3 illustrates the sorting procedure.

Table 5.5 Distribution of Trip Demand Percentage

OD Pairs	Trip Demand	Percentage
Largest 500	480,343	52.9%
Largest 1,000	629,486	69.3%
Largest 1,500	714,038	78.6%
Largest 2,000	769,650	84.7%
Largest 2,500	809,697	89.1%
Largest 3,000	839,399	92.4%
Largest 4,000	861,198	94.8%
Largest 4,500	877,332	96.6%
Largest 5,000	889,148	97.9%
Largest 5,500	897,715	98.8%
Largest 6,000	903,644	99.5%
All 6,561	908,281	100%

The first 1,000 OD pairs carry 69.3% of the total travel demand in the study area. It is assumed that this large amount of trips would be sufficient in identifying irrelevant links.

Meanwhile, running the path-tracing model for 1,000 OD pairs would reduce the runtime of the path-tracing model to about 1.5 days.

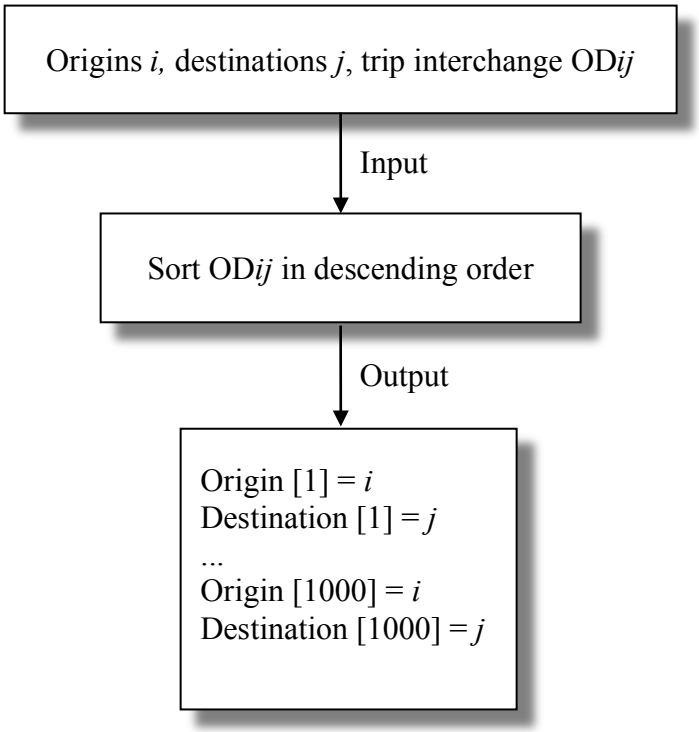

Figure 5.3 Procedure of sorting OD trip interchange

5.3.2 Path-tracing Model

The path-tracing model begins with the OD pair with the largest trip interchange. The trip assignment program assigns the trip table on the entire network, while keeping track of links used by this selected OD pair. This procedure is repeated 1,000 times to record the path-tracing results for the selected 1,000 OD pairs. Figure 5.4 presents the trip assignment result for the entire network with the path-tracing result for the 1st largest OD pair.

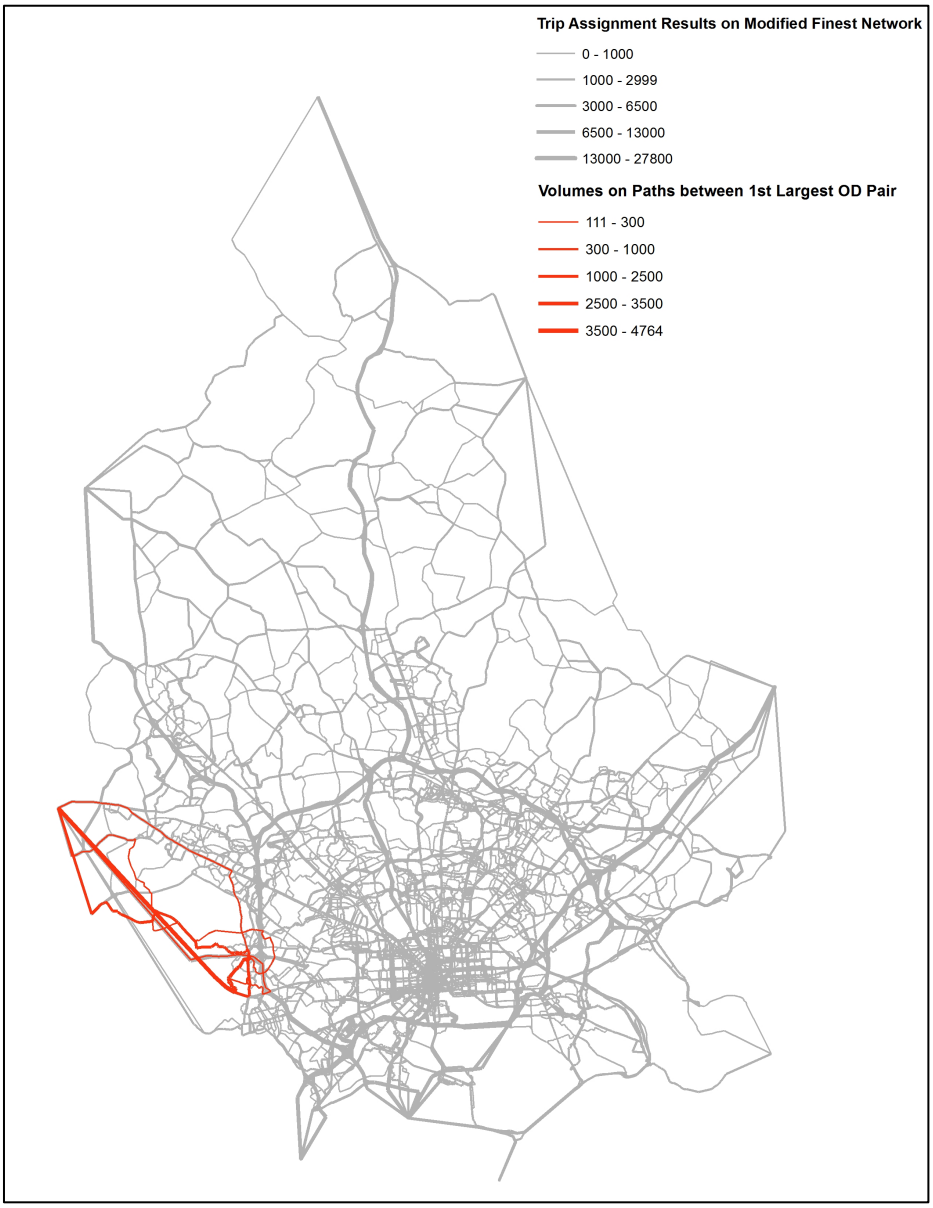

Figure 5.4 Trip assignment result and path-tracing result for the 1st largest OD pair

For each of the selected OD pairs, there is a separate output file containing link numbers and volumes on the path(s) built between that specific OD pair. Table 5.6 shows the path-tracing result for the 1st largest OD pair on the modified finest network.

Table 5.6 Path-tracing Result for Origin Zone 64 and Destination Zone 79

Start Node_End Node	Start Node	End Node	Link Volume
64_7060	64	7060	1551

64_7128	64	7128	3324
64_7407	64	7407	1773
...	...	...	...
1002_79	1002	79	111
1004_79	1004	79	111

5.3.3 Identify and Remove Irrelevant Links

A Cube script was developed to index links on the current network. For each OD pair, a Java program reads in its path-tracing result to calculate the trip shares for links belonging to this OD pair. In particular, a link-volume matrix (two-dimensional arrays) is created in Java to hold and analyze the path-tracing results. The link index and sorted OD array (by trip interchange) represent the row and column numbers in the link-volume matrix, as presented in Table 5.7.

Table 5.7 Link-volume Matrix Holding Path-tracing Results

	1st OD	**2nd OD**	**...**	**999th OD**	**1000th OD**
Link 1	$Vol_{1,1}$	$Vol_{1,2}$	...	$Vol_{1,999}$	$Vol_{1,1000}$
Link2	$Vol_{2,1}$	$Vol_{2,2}$	...	$Vol_{2,999}$	$Vol_{1,1000}$
...	...	...	...	...	...
Link 48338	$Vol_{48338,1}$	$Vol_{48338,2}$	...	...	$Vol_{48338,1000}$
Link 48339	$Vol_{48339,1}$	$Vol_{48339,2}$	...	$Vol_{48339,999}$	$Vol_{48339,1000}$

After that, each element in Table 5.7 is divided by the trip interchange of corresponding OD pair to calculate a trip-share matrix, as presented in Table 5.8. A Java program was developed to analyze the trip share results for the current network and identify irrelevant links to be removed.

Table 5.8 Trip-share Matrix

	1st OD	**2nd OD**	**...**	**999th OD**	**1000th OD**
Link 1	$TR_{1,1}$	$TR_{1,2}$	...	$TR_{1,999}$	$TR_{1,1000}$
Link2	$TR_{2,1}$	$TR_{2,2}$	...	$TR_{2,999}$	$TR_{1,1000}$
...	...	...	...	...	...
Link 48338	$TR_{48338,1}$	$TR_{48338,2}$	...	...	$TR_{48338,1000}$
Link 48339	$TR_{48339,1}$	$TR_{48339,2}$	...	$TR_{48339,999}$	$TR_{48339,1000}$

In this study, the percentage of total number of links is the primary parameter to identify irrelevant links. It represents that the total number of irrelevant links should be no more than the defined percentage of total number of links on the current network. When the program identifies the number of irrelevant links, it concurrently identifies their trip share cut-off point, the value of which represents that the irrelevant links carry no more than this amount of trip share for any of the OD pairs on the current network.

To get the Java program started, an initial value of 0.1 is set for the percentage of total number of links. It can be increased or reduced depending on the network's sensitivity to the removal of irrelevant links. The trip share cut-off point evaluates how important the irrelevant links are in relation to the network, and it can be used as an alternative of the primary parameter to identify irrelevant links.

To prevent the program from removing important links for OD pairs other than the selected 1,000 OD pairs, another parameter – the link volume-over-capacity (v/c) ratio is applied to the Java program. The link v/c ratio is calculated as dividing its assigned volume by its capacity. If it were computationally feasible to run this program with all of the OD pairs, the link v/c ratio would not be necessarily applied.

5.4 Network-defining Models

The network-defining model is developed following the methodology given in Section 4.1. It is a composite model including a path-tracing component developed in Cube, a link-identifying component developed in Java, a link-removing component developed in Cube, and a validation component developed in Java. There are several variables involved in the model implementation, and each of them plays a part in determining the final solution. In this section, different variables will be experimented and results will be compared.

5.4.1 Results of a Provisional Model

Given the limited computing power, a provisional model was developed first as a proof-of-concept of the proposed methodology. It is expected that the network defined by the provisional model may not be the most accurate one, but it can narrow down the solution to a smaller scope. The development of the provisional model could also provide an estimate of the model runtime.

Figure 5.5 presents a flowchart, illustrating how does the provisional model read and export results as well as identify and remove irrelevant links during each iteration of the model. The programs executed in Cube are denoted a [C], and programs executed in Java are denoted a [J]. The main parameters applied in each of the iterations are displayed in the flowchart. The control parameter – link v/c ratio, is defined as 0.5 to keep important links from being removed in the provisional model. In the 9^{th} iteration, validation results start to worsen (see Table 5.9), suggesting that the ideal network resolution has been reached before the 9^{th} iteration

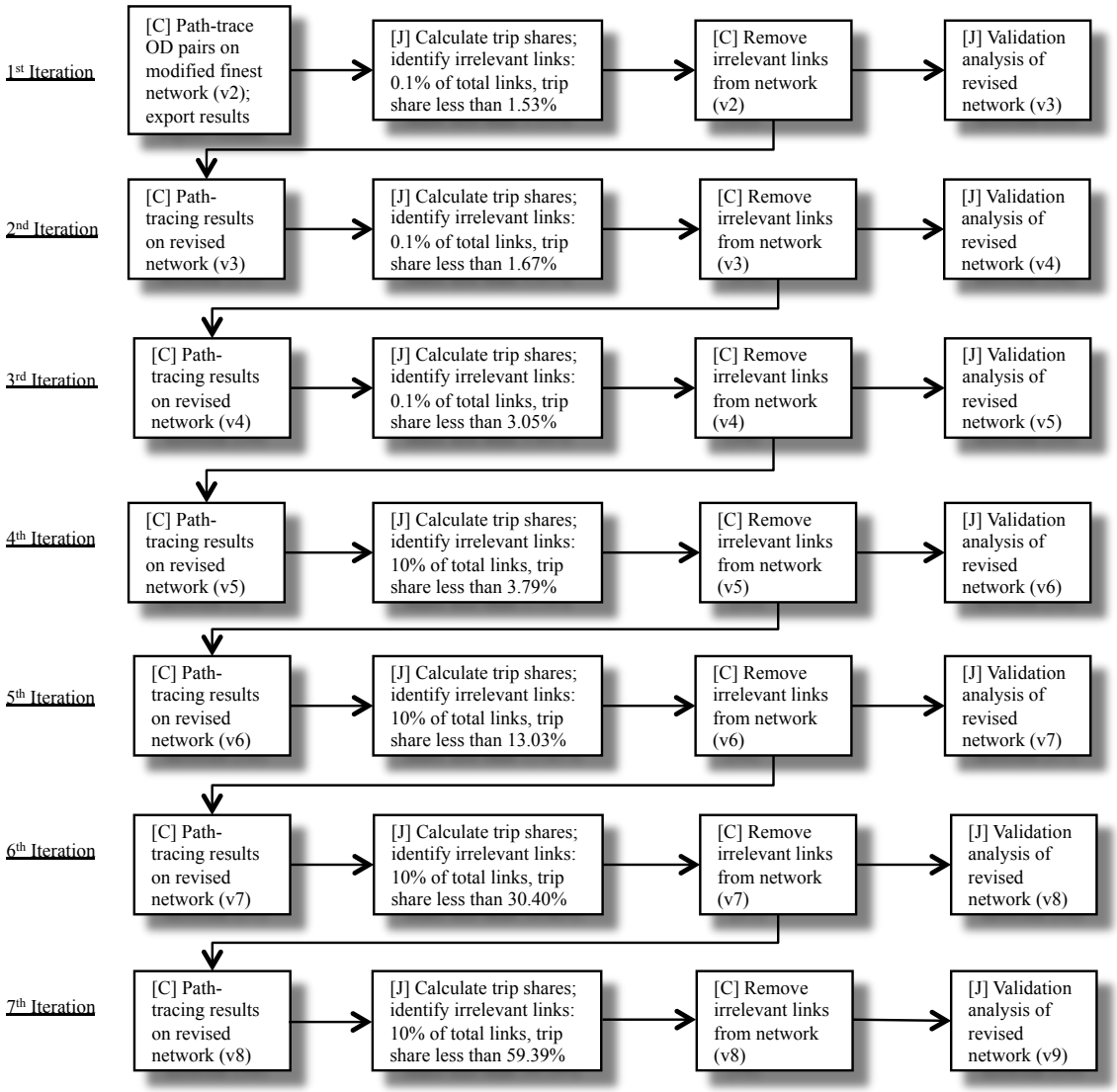

Figure 5.5 Decision-making process of identifying the ideal network resolution

As presented in the flowchart above, the provisional model starts with the modified finest network, and identifies a small number of irrelevant links for the first three iterations to test the model's sensitivity. The slowly decreasing %RMSE and improving %Difference suggest that, the network resolution is becoming more consistent with the given zone system. To accelerate the process, starting from the 5th iteration, the number of irrelevant links to be removed is increased to 10% of total links on the present network. At the 6th

iteration, the validation analysis reaches its best result so far, and it begins to decline from the 7th iteration. Table 5.9 shows model parameters and validation results for each of the model iterations. The convergence level of the provisional model was set at relative gap = 0.02, and it took about 35 hours to compute each of the model iterations.

In Table 5.9, performance measures are calcualted to evaluate the agreement between modeled volumes and observed traffic counts. These measures include the R^2, RMSE, %RMSE, %Difference, Average Error, and Difference of Standard Deviation (see Section 4.3.6). Characteristics of the current network are reported, including the total number of links and zero-volume links. Last two columns represent the parameters used to identify the irrelevant links.

Table 5.9 Parameters and Validation Analysis of the Provisional Model

V	Validation Results						Current Network Characteristics		Parameters of Identifying Irrelevant Link	
	R^2	RMSE	% RMSE	% Diff	Ave. Error	Diff. of Std. Dev.	Total Links	Zero-vol Links	% Total Links	Trip-share
1	0.86702	1861.6380	42.82%	-2.70%	-120.59	-360.88	135,364	85,082	-	-
2	0.87023	1847.3933	42.59%	-2.21%	-101.83	-399.73	48,339	0	0.1%	1.53%
3	0.87065	1844.6211	42.53%	-2.21%	-101.88	-407.46	48,293	119	0.1%	1.67%
4	0.87025	1846.8487	42.47%	-1.95%	-90.00	-404.60	48,145	181	0.1%	3.05%
5	0.87038	1846.5262	42.52%	-2.09%	-96.35	-416.43	47,927	27	10%	3.79%
6	0.87270	1835.8977	41.90%	-0.43%	-20.00	-402.67	43,115	1857	10%	13.03%
7	0.86750	1877.2679	42.71%	0.26%	12.44	-322.20	37,170	1000	10%	30.40%
8	0.87256	1894.0751	43.27%	4.02%	195.27	-165.10	32,589	856	10%	59.39%
9	0.86343	2020.6505	44.65%	9.81%	488.32	-160.06	28,592	1679	STOP	-

Network v1 is the finest network and network v2 is the modified finest network, in which all links that received no volume in the initial assignment were removed (see Section

5.2.3). From the validation results, several observations can be made: (1) it is consistent with expectations that removing irrelevant links from a finer network can improve the zone-network consistency to a certain degree; (2) when removing links from the network, the modeling results first underestimate traffic conditions but gradually move forward to an overestimation; and (3) removing 10% of total links seems to be a big leap between the 5th and 6th iterations and between the 6th and 7th iterations, as it is very likely that the ideal network is somewhere between the 5th and 7th iterations.

5.4.2 Zero-volume Links after Removing Irrelevant Links

It can be seen from Table 5.9 that removing irrelevant links can lead to a number of zero-volume links on the network. There seems to have a linear relationship between the number of irrelevant links identified and the number of zero-volume links. The zero-volume links could be a result of redistributing traffic flows on a smaller network, or it could be due to network discontinuity. Therefore, it is necessary to examine the cause of zero-volume links to make sure that there is no discontinuity issue.

For instance, after removing 0.1% of total links as irrelevant links from network v2, there are 119 zero-volume links on network v3. Figure 5.6 presents some zero-volume links on network v3. The highlighted links had 10 vehicles assigned on network v2 but became unused on network v3. Meanwhile, this path is correctly connected to the network after checking its connectivity with adjacent links. Obviously, these 10 vehicles were rerouted to other paths on network v3, which could reduce users' travel time. After conducting the same visual inspection on all other zero-volume links, it can be concluded that the MSA algorithm redistributed traffic flows on the new network, therefore leaving

some links unused. Furthermore, since a looser convergence level was applied in the provisional model, the assignment results may not be equilibrium solutions.

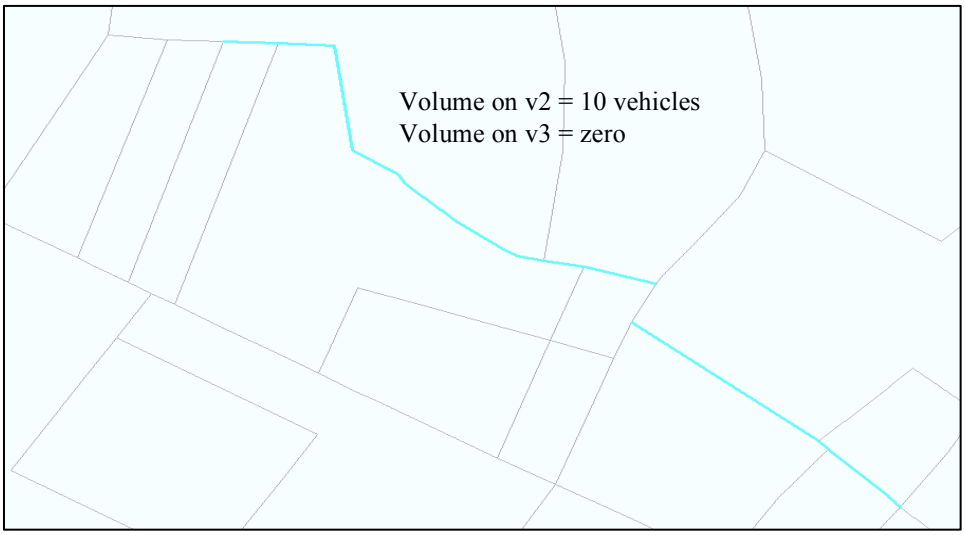

Figure 5.6 Example of zero-volume links on network v3

If looking at zero-volume links on network v6, which is the product after removing 10% of total links on network v5, it is found that several zero-volume links are dangling links. Figure 5.7 demonstrates an example of a dangling zero-volume link, which carried 12 vehicles on network v5. This link became a dangling link after its connecting links were removed as irrelevant links from network v5. Because only the most important 1,000 OD pairs were considered in identifying irrelevant links in the provisional model, it is possible that the dangling link was not used by any of the 1,000 OD pairs. If more than 1,000 OD pairs were included in the analysis, it is assumed that the number of dangling links would be smaller. However, due to the limited computing power, applying all of the OD pairs were not feasible for the time being.

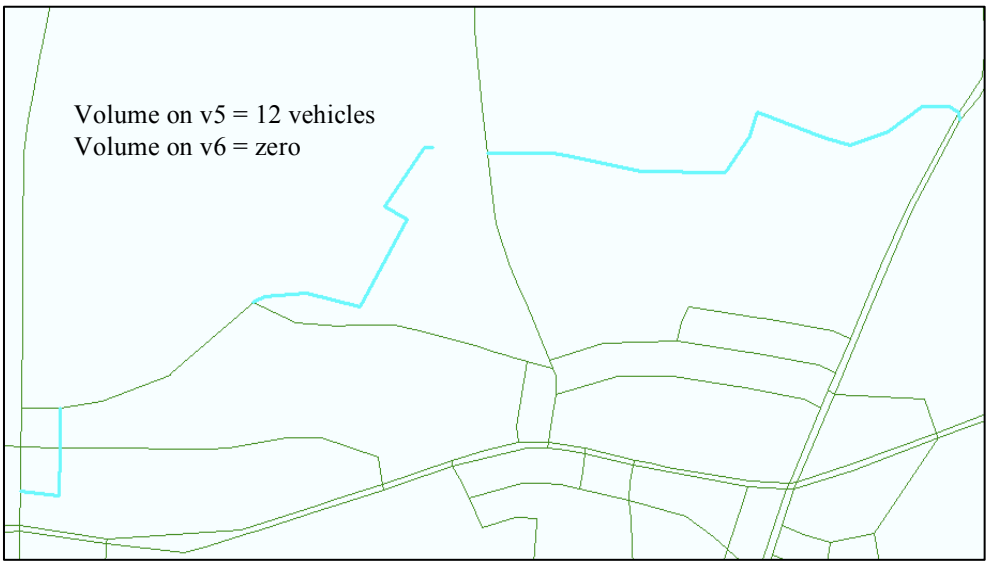

Figure 5.7 Example of a dangling zero-volume link on network v6

5.4.3 Smaller Steps of Irrelevant Link Removal

Considering that the model is sensitive to the number of irrelevant links removed, removing a large portion of links during each of the model iterations may not lead to a precise solution. It is necessary to apply a smaller variable to identify and remove irrelevant links. Table 5.10 presents the results of the provisional model after applying a smaller removal step from the 5th iteration on. This model starts from network v5 in Table 5.9.

Table 5.10 Parameters and Validation Analysis of the Smaller-Step-Model

V	Validation Results						Current Network Characteristics		Parameters of Identifying Irrelevant Link	
	R^2	RMSE	% RMSE	% Diff	Ave. Error	Diff. of Std. Dev.	Total Links	Zero-vol Links	% Total Links	Trip-share
5	0.87038	1846.5262	42.52%	-2.09%	-96.35	-416.43	47,927	27	5%	3.26%
5.1	0.87255	1835.2628	41.97%	-0.87%	-40.68	-419.49	45,544	1220	5%	3.58%
5.2	0.87282	1834.9602	41.88%	-0.42%	-19.62	-400.59	42,175	834	5%	6.44%
5.3	0.87071	1855.7071	42.43%	-0.05%	-2.24	-366.21	39,379	438	2%	3.23%

5.4	0.87263	1842.5307	42.04%	0.17%	7.87	-315.26	38,172	244	STOP	-

After comparing the results in Table 5.9 and Table 5.10, it can be seen that the best networks calculated by the two approaches have reached similar resolution (zero-volume links excluded). In Table 5.9, the optimal network is v6 with 41,258 links; and in Table 5.10, the optimal network is v5.2 with 41,341 links. Comparing their validation results, network v5.2 is slightly better than network v6. The findings suggest that the order of removing irrelevant links may not play an important role in determining the optimal network resolution. However, removing irrelevant links in smaller steps can help locate a more consistent network resolution for a given zone system.

5.4.4 Larger Coverage of OD Pairs

In this section, the most important 2,000 OD pairs, which accounts for almost 85% of the travel demand in the study area, are applied. The model results, as presented in Table 5.11, will be compared with networks developed by the provisional model, which analyzed the most 1,000 OD pairs. Since there are still a number of OD pairs that are not taken into analysis, the control parameter v/c ratio of 0.5 is still in use to prevent the model from removing important links for unselected OD pairs. The runtime for each of the model iterations takes about 70 hours, which is as twice as that of running an iteration with 1,000 OD pairs.

The model starts with the modified finest network and proceeds by removing 5% of total links as irrelevant links. When the improvement of validation results begins to slow down, it suggests that the resolution of the current network is close to the optimal one. At this point, it is necessary to start applying a smaller step to the link removal,

which is network v4 in Table 5.11. The validation results continue to improve until network v5, and additional efforts are made for the final iteration on network v5 (see Table 5.12). Links are removed from v5 in 1-percent increments (1%, 2%, 3%,…, 10%) to gradually narrow down to the optimal network resolution.

Table 5.11 Parameters and Validation Analysis of the Larger-Coverage-Model

V	Validation Results						Current Network Characteristics		Parameters of Identifying Irrelevant Link	
	R^2	RMSE	% RMSE	% Diff	Ave. Error	Diff. of Std. Dev.	Total Links	Zero-vol Links	% Total Links	Trip-share
1	0.86702	1861.6380	42.82%	-2.70%	-120.59	-360.88	135,364	85,082	-	-
2	0.87236	1830.7008	41.98%	-1.67%	-76.94	-373.22	48,339	70	5%	3.12%
3	0.87396	1822.5717	41.72%	-1.20%	-55.80	-400.07	45,930	637	5%	3.64%
4	0.87667	1805.3509	41.34%	-0.97%	-45.21	-401.72	43,076	257	2%	4.60%
5	0.87622	1807.5112	41.14%	-0.37%	-17.12	-376.06	41,963	87	see Table 5.12	

For networks v5 to v6.10, the negative values of Percent Difference, Average Error and Difference of Std. Dev. imply that, these networks all underestimate the traffic conditions to some extent. This is in part due to the removal of some links from validation analysis. Another reason could be, the assigned volumes on the removed links have been redistributed to other routes, the effect of which is not captured on the links used for validation analysis. After removing 3% of total links from network 5, network v6.3 in Table 5.12 has the best validation results compared to the other nine networks, and therefore it is selected as the optimal network.

There are several interesting findings when comparing network v6.3 in Table 5.11 with network v5.2 in Table 5.10. First, network v6.3 has fewer links (N = 40,514) than network v5.2 (N = 41,341) but with improved validation results. This finding implies that

the resolution of network v6.3 is more consistent with the resolution of the zone system. Second, the optimal networks developed by the two models have similar degree of agreement with the observed traffic counts. However, running the model with the most important 2,000 OD pairs takes twice the time of running a model with the most important 1,000 OD pairs. Given the limited computing power, running the model with 1,000 OD pairs would be sufficient to achieve a desirable network resolution.

Table 5.12 Parameter Testing on Last Iteration of Network V5

V	Validation Results						Current Network Characteristics	
	R^2	RMSE	% RMSE	% Diff	Ave. Error	Diff. of Std. Dev.	Total Links	Zero-vol Links
5	0.87622	1807.5112	41.14%	-0.37%	-17.12	-376.06	41,963	87
Test 1: Remove 1% Total Links with Trip-share of 6.37%								
6.1	0.87567	1813.6637	41.35%	-0.38%	-17.92	-398.21	41,531	38
Test 2: Remove 2% Total Links with Trip-share of 6.61%								
6.2	0.87560	1815.1927	41.39%	-0.33%	-15.55	-409.31	41,039	100
Test 3: Remove 3% Total Links with Trip-share of 8.15%								
6.3	0.87622	1810.9483	41.25%	-0.22%	-10.31	-417.97	40,623	109
Test 4: Remove 4% Total Links with Trip-share of 9.77%								
6.4	0.87525	1817.8093	41.44%	-0.30%	-14.22	-412.84	40,229	115
Test 5: Remove 5% Total Links with Trip-share of 11.98%								
6.5	0.87358	1829.4401	41.59%	-0.03%	-1.44	-400.18	39,817	105
Test 6: Remove 6% Total Links with Trip-share of 13.00%								
6.6	0.87433	1824.2597	41.60%	-0.34%	-16.31	-406.66	39,377	155
Test 7: Remove 7% Total Links with Trip-share of 16.09%								
6.7	0.87578	1815.4524	41.46%	-0.27%	-12.91	-387.09	38,978	187
Test 8: Remove 8% Total Links with Trip-share of 16.38%								
6.8	0.87462	1824.3214	41.74%	-0.46%	-21.51	-407.12	38,532	208
Test 9: Remove 9% Total Links with Trip-share of 19.43%								
6.9	0.87410	1828.6043	41.73%	-0.21%	-9.95	-428.85	38,116	213
Test 10: Remove 10% Total Links with Trip-share of 22.58%								
6.10	0.87417	1831.1351	42.00%	-0.34%	-16.22	-430.99	37,696	235

Given all the networks developed so far, network v6.3 has the most appropriate resolution for the given zone system. Therefore, this network will be used as the final network in the following analyses.

5.5 Analyses of Finest Network and Final Network

Given the finest network and the final network developed in previous sections, a user-equilibrium solution is computed by the MSA assignment on both networks by applying the recommended convergence level of relative gap = 0.0001. Comparison analyses will be conducted to illustrate the improved modeling results on the final network.

5.5.1 Comparison of Modeled Volumes – Individual Links

The frequency distributions of link flow rate at the interstate highway level as well as at the freeway and expressway level are examined. The link flow rate is calculated as vehicle per hour per lane (VPHPL).

Interstate Highway

Figure 5.8 shows the flow rate distributions of interstate highway links (functional class = 1) on the finest and final networks. Overall, same tendency can be observed on both networks: a large amount of links is clustered around 1,500 vphpl, and a fairly small group of links distributes at two tails. Differences can be observed as well. For instance, on the final network, there is a slight increase between 1,300 – 1,400 vphpl and between 1,800 – 2,000 vphpl.

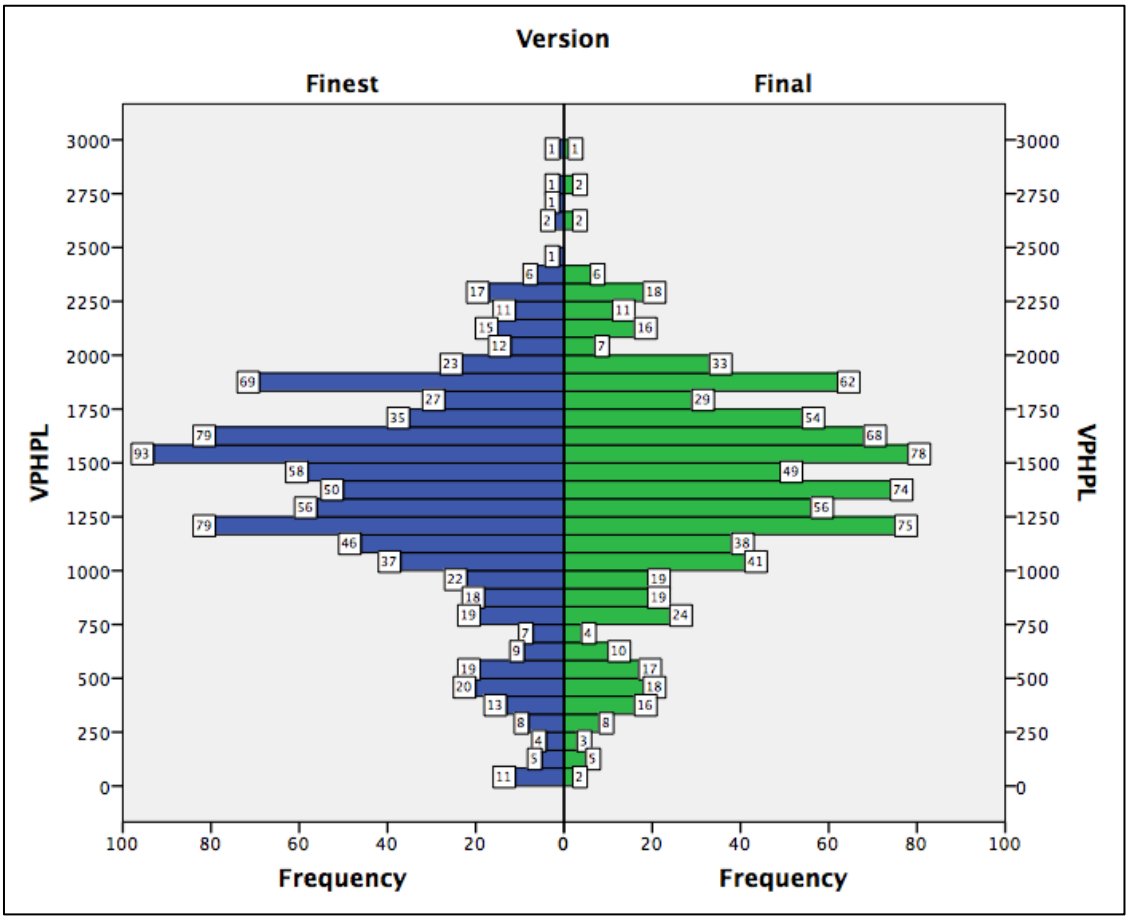

Figure 5.8 Link flow rate distributions of interstate highway links

Freeway and Expressway

Figure 5.9 shows the flow rate distributions of freeway and expressway links (functional class = 2). Overall, the distributions on both networks are nearly equal. The final network just has a few more heavy-load links than the finest network.

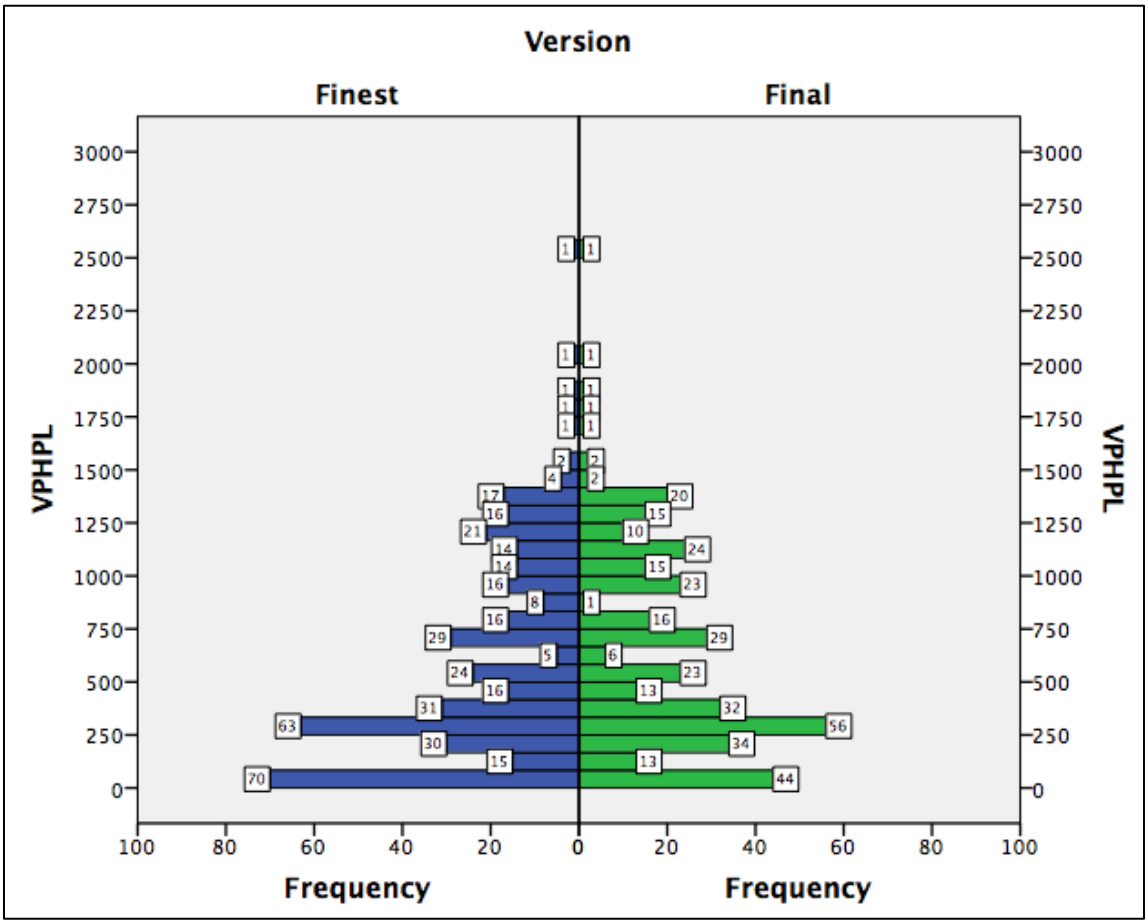

Figure 5.9 Link flow rate distributions of freeway and expressway links

From the analyses above, it is obvious that the flow rate distributions of interstate highway as well as freeway and expressways do not change significantly between the final and finest networks.

5.5.2 Comparison of Modeled Volumes – Entire Network

Vehicle Miles Traveled (VMT) and Vehicle Hours Traveled (VHT) are important measurements of the magnitude of the overall traffic volumes. To calculate the VMT for the current network, the product of the number of vehicles on every link is multiplied by the length of the link as link VMT, then all link VMTs on the network are summed up as

the total VMT. The VHT is calculated by multiplying the travel time and the number of vehicles on every link and summing it up for all links on the network.

Table 5.13 VMT and VHT on Finest and Final Networks

Functional Class	VMT			VHT		
	Finest	Final	% Change	Finest	Final	% Change
Interstate	2,870,091	2,880,258	0.35%	70,695	70,903	0.29%
Freeways and Expressways	293,903	290,159	-1.27%	8,997	8,880	-1.30%
Major Arterial	1,388,310	1,405,745	1.26%	65,548	67,564	3.08%
Minor Arterial	945,689	941,698	-0.42%	51,925	50,919	-1.94%
Major Collector	559,832	557,443	-0.43%	58,625	62,511	6.63%
Minor Collector	94,841	91,804	-3.20%	4,398	4,213	-4.21%
Local	1,171,515	1,165,687	-0.50%	106,666	107,046	0.36%
Total	9,058,755	9,074,768	0.18%	464,750	470,310	1.20%

Several observations can be made from Table 5.13. First, interstate highway links and principal arterial links gained more VMT and VHT on the final network. However, it is unexpected for the freeways and expressway links to have less VMT and VHT on the final network, because freeways and expressways are supposed to take some volumes from the removed links. One explanation is that the relatively small amount of freeway and expressway links is very sensitive to the change made on these roads. Actually, 13% (55 out of 438) of the freeway and expressway links were removed from the finest network due to their very small trip shares, and the change in the number of links has resulted in a shift of volumes between routes accordingly. The lower class links all experienced a loss of VMT or VHT on the final network. It is reasonable to have declined VMT and VHT on minor roads and collectors, because many of these roads were removed as irrelevant links.

5.5.3 Comparison of Validation Results

Further insights into the improvement of the final network can be gained by comparing molded volumes with observed traffic counts on a link-by-link basis. There are 210 directional count stations in the present study. The number of links with assigned volumes varies with the changes in network resolution. The validation results are conducted for links that have assigned volumes and observed counts. The analysis results are presented for all links and by functional class.

Table 5.14 Volume-count Agreement on Finest and Final Networks

Network	Link Group	N*	Validation Results					
			R^2	RMSE	% RMSE	% Diff	Ave. Error	Diff. of Std. Dev.
Finest	All	204	0.865599	1880.7360	43.70%	-3.11%	-142.21	-417.52
	Interstate	25	0.384376	3746.4725	24.14%	-7.64%	-1283.64	-632.08
	Freeway/ Expressway	16	0.550139	2250.0237	40.70%	-11.49%	-717.50	-611.72
	Major Arterial	114	0.256253	1477.5869	48.36%	5.00%	145.38	489.45
	Minor Arterial	38	0.223629	1053.4703	75.56%	-1.09%	-16.74	-1.68
Final	All	197	0.874648	1822.7150	41.96%	-1.27%	-59.57	-401.42
	Interstate	25	0.423653	3574.8484	22.76%	-6.52%	-1095.32	-539.59
	Freeway/ Expressway	16	0.547120	2258.1611	40.86%	-11.52%	-719.56	-610.36
	Major Arterial	114	0.314003	1433.6000	46.55%	8.04%	237.43	413.91
	Minor Arterial	38	0.250481	1079.3571	75.58%	3.39%	53.88	39.40

* links with functional class lower than Minor Arterial are excluded from validation analysis due to a very small sample size

By comparing the overall validation results between the finest and final networks, the better agreement with observed traffic counts on the final network indicates an improved network-zone consistency. According to Table 5.14, trips assigned on interstate

highways, freeways and expressways are underestimated in general, but the degree of underestimation is lower on the final network. On the contrary, major arterial and minor arterial links carry more assigned volumes than the observed traffic counts.

Taking the results in Table 5.13 and Table 5.14 together, the changes in the validation results between the finest and final networks are consistent with expectation. On the final network, interstate highway links and major arterial links were assigned more volumes, taking away traffic from the removed lower class links, which improves the overall validation results. No substantial changes can be observed for a small number of freeway and expressway links (N = 16). The minor arterial links actually experience a loss of VMT and VHT on the final network, according to the results in Table 5.13. However, the results in Table 5.14 reveal that the minor arterial links overestimate the traffic conditions. Again, the small sample size of the validation links may confound the validation results.

5.5.4 A Case Study on Final and Finest Networks

Findings from previous sections suggest that it is necessary to closely examine the changes in assigned volumes in areas where massive irrelevant links are removed. Figure 5.10 presents a case study for zone 62, looking into the details on the final and finest networks. Links from five functional classes are closely examined, including I-95 (interstate highway), Belair Road (principal arterial), Rossville Road (minor arterial), Perry Hall Boulevard (minor arterial), King Avenue (major collector), and Linda Avenue (local road).

After removing irrelevant links from the finest network, the I-95 segment shows a very small increase in its assigned volumes in the northbound direction and a slight decline in the southbound direction. A decrease in assigned volumes is also found in the westbound Linda Avenue, eastbound King Avenue and southbound Perry Hall Boulevard, but the differences between the final and finest volumes are insignificant (by 1%). These marginal changes indicate that these links are not very sensitive to the changes made on the network resolution. Especially for the interstate highway links, they do not receive a lot of volumes from the removed links. Furthermore, on the final network, a significant increase in assigned volumes can be observed on the Belair Road and Rossville Road in both directions, Perry Hall Boulevard and Linda Avenue in the northbound direction, and King Avenue in the westbound direction. It is obvious that these roads have received additional volumes from removed links. Considering the relatively small volumes on the removed links, it is reasonable to have a small volume growth on some of these roads.

A closer inspection of Figure 5.10 reveals an important finding. For major collector, minor arterial and local roads, it is more likely for them to absorb volumes from removed links than the higher class links.

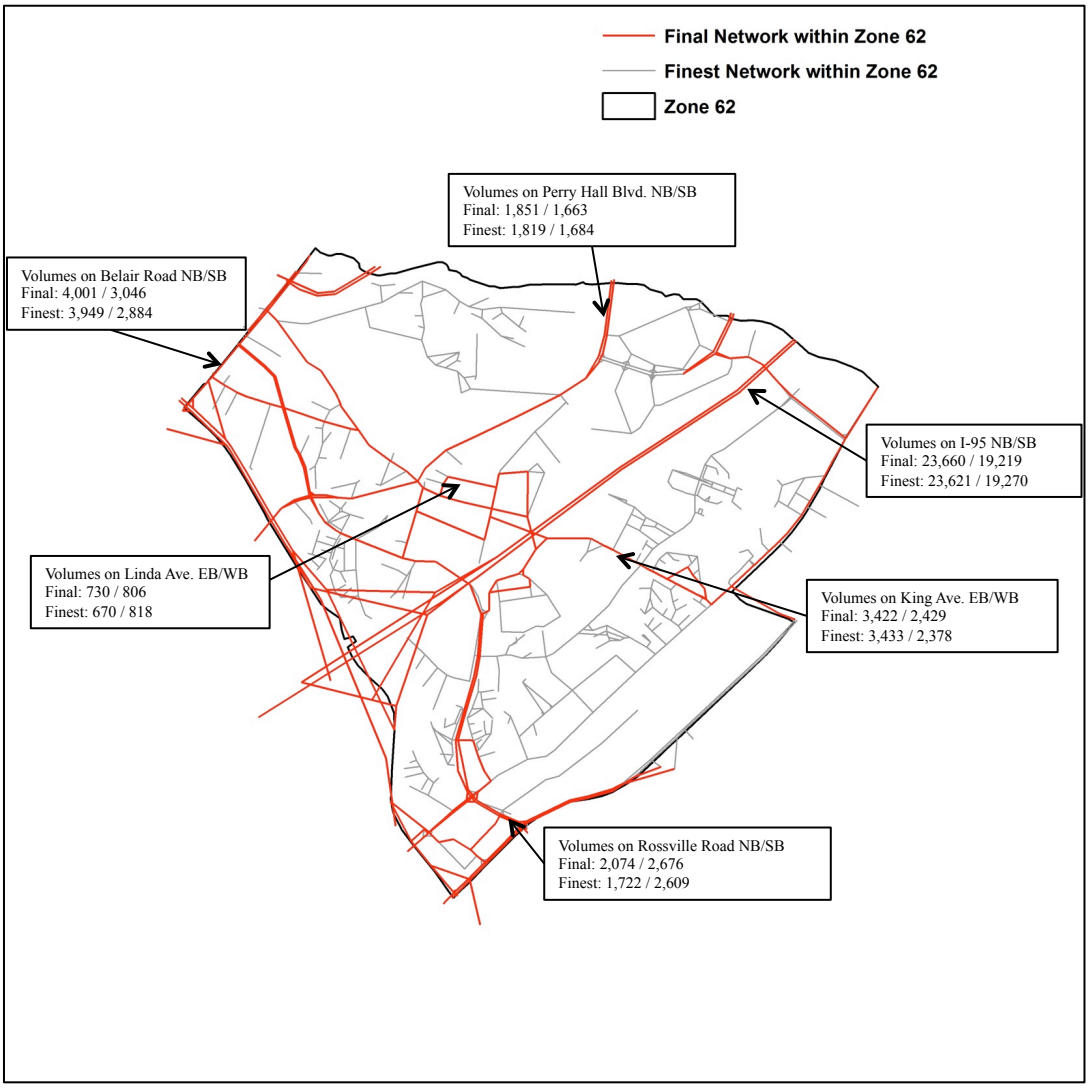

Figure 5.10 Comparisons between the final and finest networks for zone 62

5.6 Implementation of a New Transportation Project

Ultimately, it is important for this dissertation to explore the ability of the final network to provide forecasts of the impacts of proposed highway projects. In this section, a proposed project is taken from the BMC 2040 long range plan and coded on the final and finest networks.

The chosen project is part of the BMC 2040 planning scenario and involves the widening of I-695 from 3 lanes to 4 lanes in each direction on the section connecting I-95

and MD 122 (23). To evaluate the impacts of this new project, a pair of model runs is compared with and without the project. This was done for both the finest network and the final network during the PM peak hour period. The traffic assignments achieved a convergence level with a relative gap = 0.0001 using the MSA algorithm. The extent and location of the I-695 widening project are shown in Figure 5.11.

.

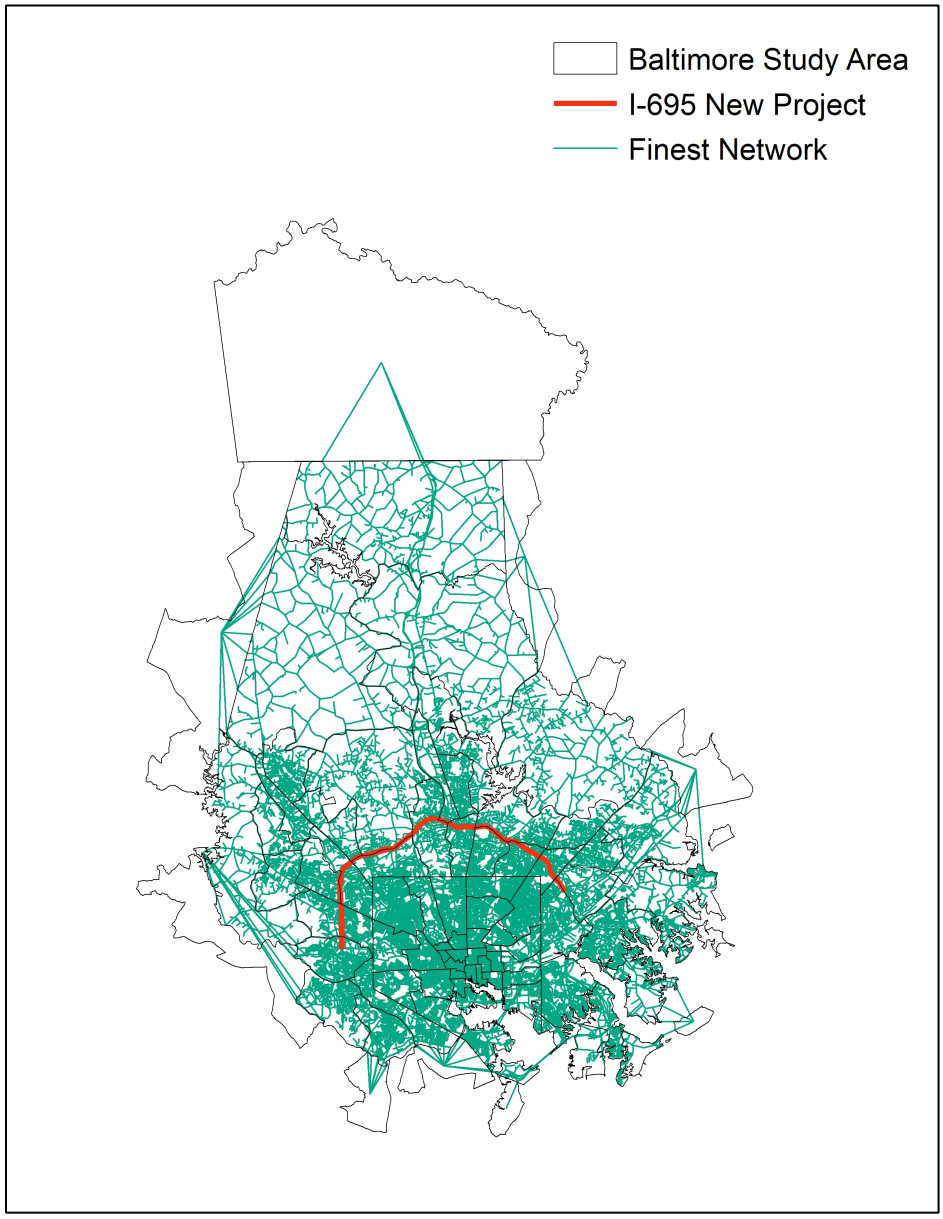

Figure 5.11 The location and extent of I-695 new project in the fines network

The impacts of the new project on the finest network and the final network are presented in Figure 5.12 and Figure 5.13. On the finest network, after adding the new project, the v/c ratio of 624 links dropped below one, while the v/c ratio of 447 links increase above one. Same patterns can be observed on the final network, where the v/c ratio on 603 links declined below one while on 410 links risen above one. The similar changes observed in the v/c ratio on both networks imply that the final network is very well connected to represent the extent of the new project's impacts.

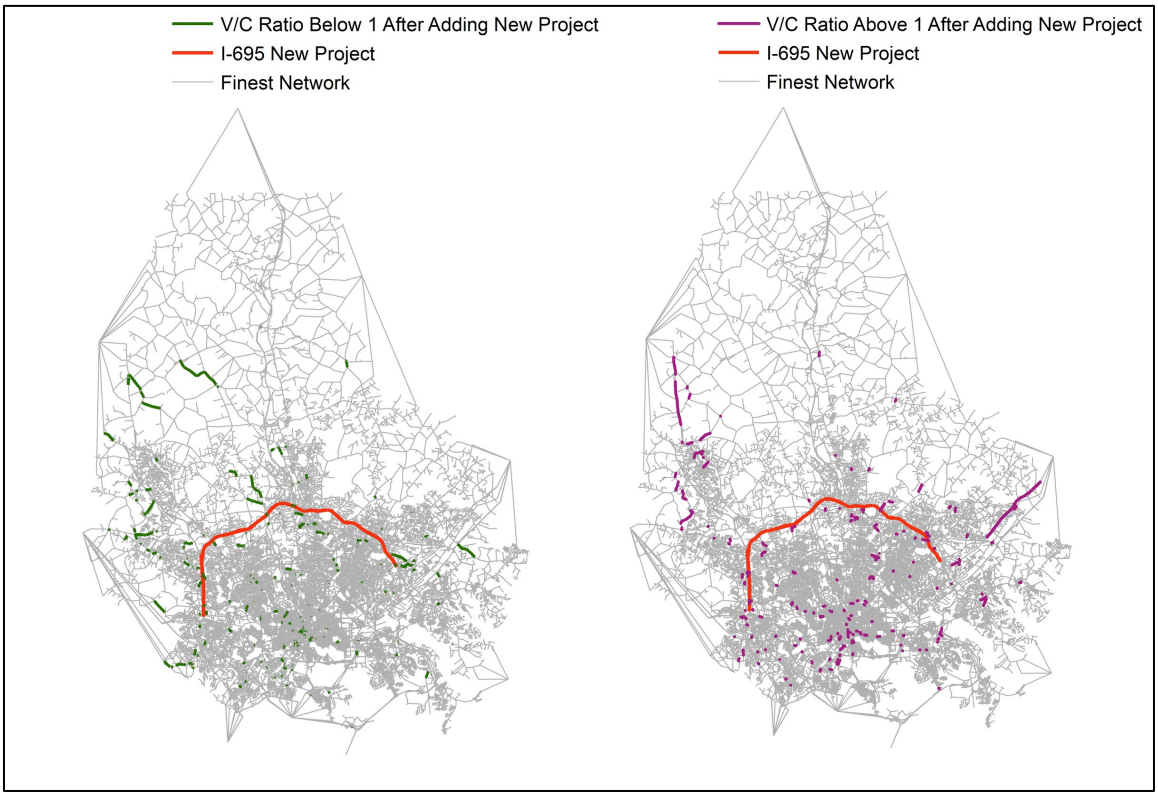

Figure 5.12 V/C ratio changes on finest network after implementing the new project

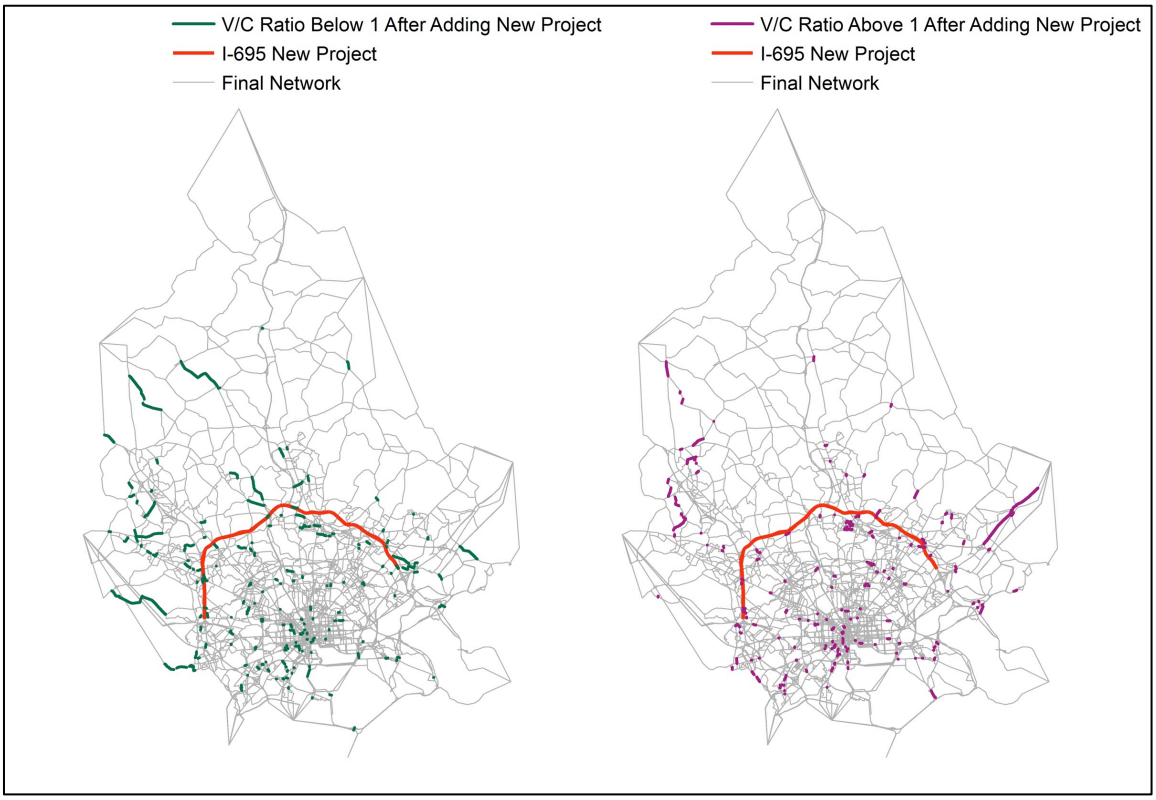

Figure 5.13 V/C ratio changes on final network after implementing the new project

More specifically, the VHT on the finest network deceased by 3.21% from 464,356 to 449,440; and the VHT on the final network declined by 3.32% from 466,118 to 450,651. The finest network underestimates the real traffic conditions to a larger degree, which makes it react not as sensitively as the final network.

When looking at the impacts of the new project on the entire Interstate Highway I-695, the final network revealed a more convincing result. On the finest network, the VMT on I-695 grew from 1,107,230 to 1,186,882, while the VMT on the final network increased from 1,111,309 to 1,187,522. The detailed finest network provides more alternative routes to road users, which makes it underestimate the VMT on the I-695 in both conditions (before and after the project implementation).

What can be concluded from these findings is that, the modeled volumes on a consistent zone-network system are much more closer to the traffic conditions in reality. When evaluating a new project before its implementation, it is desirable to have a model that can reproduce the actual traffic volumes as much as possible. From the analyses above, it is evident that that the finest network underestimates the VHT on the entire network and the VMT on the interstate highway, where the new project is going to be implemented. As for the final network, its network resolution is more consistent with the zone system; therefore its predictions of the new project's impacts should be more accurate. A more accurate modeling result can further lead to improved planning decisions, especially when it comes to the implementation of new projects.

6 CONCLUSIONS AND IMPLICATIONS

6.1 Summary of Research and Findings

The quest for more accurate modeling results has been one of the central problems in the development of a travel demand model, and is certainly continuing to be. The consistency between the spatial resolutions of the network and zone systems plays an important role in influencing the modeling accuracy. This research answers two of the basic questions: how to evaluate the consistency of the zone and network systems? How to define the spatial resolution of the network system in consistency with a given zone system?

Some progress has been made in answering the first problem. Evidence found in empirical studies has proven that consistent network and zone systems can improve the agreement between modeled volumes and observed traffic data. However, no method in defining consistent network and zone systems has been provided. The second problem mentioned above is more important but rarely researched. A body of literature has focused on aggregating network resolution in order to reduce computation time, but little attention has been given to the impact of zone-network consistency on modeling results.

This dissertation set out to develop a network-defining model, which defines the spatial resolution of the network system in consistency with a given zone system. This model determines the zone-network consistency by evaluating the degree of agreement between modeled volumes and observed traffic counts. Several data sources and a calibrated travel demand model were prepared for the model implementation.

The model implementation was carried out in the Baltimore area, including the Baltimore County and Baltimore City. The implementation has experimented three models:

(1) A provisional model (Model I), which analyzed the largest 1,000 OD pairs. For each iteration of the model, it identified and removed 10% of the total links, which had lowest link trip shares on the current network.

(2) A revised version of the provisional model (Model II), which identified and removed 5% of the total links for each iteration of the model.

(3) A Model III, which analyzed the largest 2,000 OD pairs. For each iteration of the model, it identified and removed no more than 5% of the total links.

The final networks developed by the three models have reached similar levels of spatial detail: Model I has 41,258 links, Model II has 41,341 links, and Model III has 40,514 links. Model III has the best validation results among the three models, followed by Model II. However, Model III's runtime (9 days) is significantly longer than the other two models (3 days for Model II and 1.5 days for Model I). There is a noticeable trade-off between modeling accuracy and computation time. If the computing resources were made unlimited, applying a smaller variable in identifying and removing irrelevant links or/and analyzing a greater number of OD pairs is recommended.

The performances of the final model (Model III) and the finest model were analyzed in three aspects:

(1) For interstate highways, freeways and expressways, frequency distributions of individual link flow rate were compared for the two models.

(2) VMT and VHT of each functional class were compared for the two models.

(3) Validation analysis was performed on the two models, assessing the agreement between modeled volumes and observed traffic counts.

Findings can be summarized as follows. First, the distributions of link flow rate on the final and finest networks are similar. This finding implies that, removing a certain amount of low trip-share links is a feasible method; it increases the consistency between the network and zone resolutions without disrupting the network connectivity on important routes. Second, compared to the finest network, the final network assigned the same amount of trips through a smaller network. Most of the volumes on the removed links were taken over by major arterial roads, as suggested by the increased VMT on these roads. Adding volumes to the remaining links led to a growth in VHT for most of the link groups. This finding suggests that the MSA assignment algorithm worked well in redistributing volumes on the final network, and it reacted sensitively to the changes made on the network resolution. Third, according to the validation analysis, modeled volumes on the final network were closer to observed traffic counts. This finding proves that refining the network resolution by removing low trip-share links improved modeling accuracy significantly. Taken these findings together, it can be concluded that this dissertation provided a feasible methodology to define the network resolution to be consistent with a given zone system.

More convincing findings can be found by evaluating the impacts of a new project on the final and finest networks. The Baltimore 2040 long range plan has proposed a new transportation project on Interstate Highway I-695. Compared to the modeling results of the final network, the finest network underestimated the congestion before and after the project implementation to a large degree. It is expected the final

network, which is more consistent with the zone system in their spatial resolutions, provided a more accurate prediction of the new transportation project's impacts in the study area. It is further expected that the final network can help planners and policymakers make improved planning decisions before project implementations.

This dissertation is associated with several limitations. First, the collected observed traffic count data are insufficient for lower class links to establish statistical analyses for them. It would be ideal to develop a validation data pool, including a sufficient amount of data for each functional class. Random samples can be drawn from this data pool to deliver statistically robust validation results. Second, different network sources were used to develop the finest network. Consolidating inconsistent network representations demanded a considerable amount of manual efforts and quality checks. If there were a network, which has perfect spatial representation of the road network in reality, not only ineffectiveness but also human errors could be avoided when developing the finest network. Third, the methodology was implemented in an economical way due to limited computing resources. When more resources were made available, it would be ideal to apply a stricter model convergence level and expand the analysis to all OD pairs.

6.2 Implications for the Development of Travel Demand Models

Previous chapters have provided insights into and solutions for how to develop the network resolution in consistency with a given zone system and presented results and analyses of the final network developed. When comparing with observed traffic counts, the final network produced reliable modeling results. These findings point out that it is critical to have consistent network and zone systems in travel demand models, especially

when planners and policymakers use the modeling results to conduct impact analysis for new planning projects or policies.

Travel demand modelers have been aware of the importance of the zone-network relationship. For example, according to the U.S. Department of Transportation: "Experience in traffic assignment application indicates that this zone-network compatibility helps ensure that assignment results will be as accurate as possible regardless of the level of detail and the objectives of the study." *(24)*. The Ohio Transportation Department recognized the importance of the zone-network consistency when developing their travel demand models, and used it for network calibration if certain areas overestimated or underestimated traffic conditions *(3)*. A recent National Cooperative Highway Research Program Report also points out: "Highway networks are developed to be consistent with the TAZ system...network coding is finer for developed areas containing small zones and coarser for less-developed areas containing larger zones." *(25)*

However, none of these reports provide a systematic solution or specific guidelines for defining spatially consistent zone and network systems. In sections below, guidelines and recommendations are provided to travel demand modelers on how to develop a network system to be consistent with the given zone system, and how to alter the current network system to be consistent with the given zone system during the stage of model calibration.

6.2.1 Develop a New Network

When developing a travel demand model, it is ideal to develop a new network for a given zone system. The type of analyses as well as other input data for the study area determine the level of detail for the zone system. After that, the highway network can be developed according to the methodology presented in this dissertation. It is desirable to develop separate networks for different time-of-day periods to include operational changes, such as reversible lanes or congestion pricing.

Specifically, it is recommended to develop the highway network following guidelines given below.

Step 1: Start from the finest possible highway network and assign basic attributes to network links, including link length, number of lanes, area type, functional class, free flow speed and capacity; other operational attributes can be considered as well, including traffic signals, reversible lanes, turning movement penalties, and toll rates. Prepare traffic count data for the same modeling year or the same time-of-day period, and match the count data with corresponding network links.

Step 2: Determine assignment method and convergence level to be used by the network-defining model, and the number of OD pairs to be analyzed in the model. By applying stringent assignment criterion and analyzing a large amount of OD pairs, it is supposed to provide more reliable modeling results. However, the trade-off between computation time and modeling accuracy should be taken into account.

Step 3: Determine the maximum number of irrelevant links to be removed, and/or the maximum link trip share of the irrelevant links. The network-defining model

identifies irrelevant links when at least one of the criteria is met. These variables can be adjusted for each iteration of the model, and it is recommended to assigned smaller values to these variables when the difference between validation results of two iterations becomes smaller. Other parameters can be applied to control the types of links to be removed, such as lower class links or links with low v/c ratio.

Step 4: By the time the validation results start to decrease, terminate the model and export the network developed in previous iteration as the final network.

6.2.2 Calibrate an Existing Network

In the stage of model calibration for a travel demand model, if the agreement between modeling results and observed traffic counts is still not desirable after calibrating all other model components, it is worth checking the spatial relationship between the network and zone systems. It is recommended to adjust the network resolution following the guidelines given below.

Step 1: Visualize the modeled volumes and traffic counts along with the network and zone systems on a map. If the model overestimates or underestimates traffic conditions in a particular area, it indicates that it is necessary to examine closely on the zone-network relationship in that area.

Step 2: For a modeling area where overestimation of traffic conditions occurs, look into the modeling results on the network links in that area. If the modeled volumes are significantly higher than observed traffic counts on most of these links, add a few links representing lower class roads in that area. Compare the modeled volumes and traffic counts again after running trip assignment on the revised network. If the additional

links alleviate the overestimation problem, add a few more links at each time until the validation results are desirable.

Step 3: For an area where underestimation of traffic conditions occurs, remove a few links representing lower class roads in that area. After running traffic assignment on the revised network, compare the modeled volumes and traffic counts again. If the eliminated links alleviate the underestimation problem, remove a few more links at each time until the validation results are desirable.

6.3 Future Direction

In concluding this dissertation, four areas are identified to direct future research. First, it is desirable to have access to different data sources, which might have a perfect network representation to develop the finest network (i.e. the OpenStreetMap). At the same time, more efforts should be spent on collecting traffic count data to support a robust validation analysis.

Second, when applying the path-tracing method in trip assignment for selected OD pairs, current models processed one OD pair at a time and repeated the same procedure as many times as the number of OD pairs to be analyzed. To make the path-tracing method more efficient, it is worth exploring an advanced approach to retrieve the path-tracing results for all OD pairs at the same time.

Third, this dissertation did not fully explore other possible methods to identify irrelevant links. In future research, the identification of irrelevant links should be further investigated using different parameters, such as link volumes or rank of link volumes on the entire network. In addition to that, it would be interesting to compare the final

networks developed by different methods and find out the most efficient and effective method to identify irrelevant links.

Finally, the ultimate goal of this research is to develop a network-defining model, which is compatible with major travel demand modeling software packages. It would be ideal to integrate this model with existing travel demand models and define the most appropriate network resolution for a given zone system as part of the modeling process.

APPENDIX A – MODEL DIRECTORY

The complete directory for the Network-defining model (with 2000 OD pairs) is presented below. * denotes the current iteration. Folders not expanded in the directory contain codes developed by a third party, or contain interim files.

```
\Models
     Run MSTM v1.0.60.EXE
     \Base_2007
     \SubareaAnalysis
          RunSubarea.bat
          BaltimoreCoCi.net
          BaltSubarea_AM.trp
          BaltSubarea_PM.trp
          subareaVehPM.trp
          subareaVehAM.trp
          MSTM_VehSub_AM.net
          MSTM_VehSub_PM.net
          zoneTransfer.s
          HwyAssignSubAreaBalt.s
          aggregateSubarea.s
          \Shapefiles
          \Print
     \NetworkDefiningModel
          \NetworkFile
          \Analysis
               linkIndex.s
               exportLinkOD.s
               validation.s
               removeLinks.s
               removeZeroLinks.s
               defAttrCC.s
               defAttrByFunc(1-9).s
               defAreaType.s
               matToOD.s
          \Assignment
               RunDissertNonZeroOneIncOD.bat
               RunDissert_Post.bat
               HwyAssign_Dissert_ODSplit.s
               HwyAssign_Dissert_Post.s
               baltNetwork_run*.net
               Dissert_Veh_PM*.net
               subareaVehPM_81zones.trp
               parameter.dat
```

```
     selectlink2000.txt
     \81zones
     \Output
     \Print
\Java
     \repository
          \javaModel
               \src
                    linkDatabase.java
                    linkRecord.java
                    sortOD.java
                    tripMatrix.java
                    validation.java
          \ openCsv
          \ commons-math-2.2-src
          \ common-base
     \data
          validationStation.csv
          Dissert_Veh_PM*.dbf
          validationLinks_*.dbf
          validationLinks_*.csv
          resultsAll.csv
          resultsInt.csv
          resultsFree.csv
          resultsMajor.csv
          resultsMinor.csv
\TraceOD
     \81zones
          selOD.csv
          excLink_*.csv
          linkIndex_*.csv
          Links*_OD(1-2000).csv
          Dissert_Veh_PM*_OD(1-2000).dbf
```

APPENDIX B – EXECUTABLE SCRIPTS AND CODES

The key Cube scripts and Java codes are attached below with comments.

Cube Script – validation.s

```
; Yuchen's Dissertation
; This script computes validate total trips and by functional class (SWFT)
; Main Directory: Models\NetworkDefiningModel\Analysis
; Yuchen Cui - 04/16/2015

 v = '5'
;network version

RUN PGM=NETWORK PRNFILE='CubePRN\Convert Network to DBF.PRN' MSG='Validation - Convert Network to DBF-2'
NETI = Assignment\Dissert_Veh_PM@v@.net
;pay attention to the file name of assignment result
LINKO = Java\data\Dissert_Veh_PM@v@.dbf
ENDRUN

RUN PGM=MATRIX MSG='Use DBISEEK to extract validation links'
FILEO RECO[1] = Java\data\validationLinks_@v@.dbf,
FIELDS= A, B, A_B, CLASS, FIPS, Station, Direction, 15_18, 16_19, Total, diff15_18, diff16_19

FILEI DBI[2] = Java\data\Dissert_Veh_PM@v@.dbf, SORT = A, B
;this file contains full link information of the entire network
;IMPORTANT: in order to use DBISeek, users need to sort all arguments to
;be used in DBISeek in the script later

FILEI DBI[1] = Java\data\validationStation.csv,
A(N) = 1, B(N) = 2, A_B(C) = 3, CLASS(N) = 4, FIPS(C) =5, Station(C) = 6, Direction(C) = 7, ThreeToSix(N) = 8, FourToSeven(N) = 9
;this file is provides the validation links to be extracted
;CSV file can be inputed as DBI, and A(N)=1 indicates the 1st data field's
;name and B(N)=2 indicates the 2nd data field's name

ZONES = 1

LOOP _K = 2, DBI.1.NUMRECORDS
;for each link to be extracted
    _READ1 = DBIREADRECORD(1, _K)
    ;get A node and B node
    myA = DI.1.A
```

```
myB = DI.1.B
myAB = DI.1.A_B
myCLASS = DI.1.CLASS
myFIPS = DI.1.FIPS
myStation = DI.1.Station
myDirection = DI.1.Direction
my1518 = DI.1.ThreeToSix
my1619 = DI.1.FourToSeven

IF (DBISEEK(2, myA, myB) == 0)
   myLink = DBISEEK(2, myA, myB)
   ;find the specific link record index in the network file
   _READ2 = DBIREADRECORD(2, myLink)
   ;read the link information in the network file
   RO.A = myA
   RO.B = myB
   RO.A_B = myAB
   RO.CLASS = myCLASS
   RO.FIPS = myFIPS
   RO.Station = myStation
   RO.Direction = myDirection
   RO.15_18 = my1518
   RO.16_19 = my1619

   RO.Total = DI.2.TOTAL_VOL
   RO.diff15_18 = DI.2.TOTAL_VOL - my1518
   RO.diff16_19 = DI.2.TOTAL_VOL - my1619

ELSE
   RO.A = myA
   RO.B = myB
   RO.A_B = myAB
   RO.CLASS = myCLASS
   RO.FIPS = myFIPS
   RO.Station = myStation
   RO.Direction = myDirection
   RO.15_18 = my1518
   RO.16_19 = my1619

   RO.Total = 0
   RO.diff15_18 = 0 - my1518
   RO.diff16_19 = 0 - my1619

ENDIF

WRITE RECO = 1
```

```
      ;write out the record
ENDLOOP

ENDRUN

RUN PGM=MATRIX MSG='Convert DBF back to CSV '
FILEI DBI[1] = Java\data\validationLinks_@v@.dbf

FILEO PRINTO[1] = Java\data\validationLinks_@v@.csv

PRINT PRINTO=1 CSV=T LIST = 'A','B','A_B','CLASS', 'FIPS', 'Station', 'Direction',
'15_18', '16_19', 'Total', 'diff15_18', 'diff16_19'

ZONES = 1

LOOP _K=1, DBI.1.NUMRECORDS
;loop over all record in the DBF file (DBI.1.NumRecords = 3)
     _myRec = DBIREADRECORD(1,_K)                    ;read first record in DBF
file
    PRINT PRINTO=1 CSV=T LIST = DI.1.A, DI.1.B, DI.1.A_B, DI.1.CLASS,
DI.1.FIPS, DI.1.Station,
    DI.1.Direction, DI.1.15_18, DI.1.16_19, DI.1.Total, DI.1.diff15_18, DI.1.diff16_19
      ;after record is read, use DI.#.fieldName to retrive the value
ENDLOOP

ENDRUN
```

Java Script – validation.java

```
/**
 * Created by Yuchen Cui on 4/16/2015.
 * The validation class is used to read validation link files and calculate validation
statistics
 */

import com.opencsv.CSVReader;
import com.opencsv.CSVWriter;
import org.apache.commons.math.stat.correlation.PearsonsCorrelation;
import org.apache.commons.math.stat.descriptive.DescriptiveStatistics;

import java.io.FileReader;
import java.io.FileWriter;
import java.util.Iterator;
import java.util.List;

public class validation {
   public static void main(String[] args) {
```

```
        //define file names
        int v = 5;
        String file = "validationLinks_"+v+".csv";

        linkDatabase station = new linkDatabase();   //for all 210 validation links
        linkDatabase station1 = new linkDatabase();  //25 interstate
        linkDatabase station2 = new linkDatabase();  //16 freeway&expressway
        linkDatabase station3 = new linkDatabase();  //114 major arterial
        linkDatabase station4 = new linkDatabase();  //38 minor arterial
        linkDatabase station5 = new linkDatabase();  //12 major collector
        linkDatabase station7 = new linkDatabase();  //3 local
        linkDatabase station8 = new linkDatabase();  //2 high-speed ramp

        //column numbers in countStation.csv
        int aNode = 0;
        int bNode = 1;
        int abNode = 2;
        int funcClass = 3;
        int fips = 4;
        //int stationID = 5;
        //int direction = 6;
        int count15_18 = 7;
        int count16_19 = 8;
        int totalVol = 9;
        int diff15_18 = 10;
        int diff16_19 = 11;

        System.out.println("Read link file: validationLinks.csv");
        try {
           CSVReader stationReader = new CSVReader(new FileReader(file));
           String [] nextStation;
           int iteration = 0; // skip first line

           while ((nextStation = stationReader.readNext()) != null) {
              if(iteration == 0) {
                 iteration++;
                 continue;
              }

              if (Math.round(Float.valueOf(nextStation[funcClass])) == 1){
                 station1.addLink(nextStation[abNode]);
           station1.addAttributes(Math.round(Float.valueOf(nextStation[count15_18])),
nextStation[abNode]);

station1.addAttributes(Math.round(Float.valueOf(nextStation[count16_19])),
nextStation[abNode]);
```

```
                station1.addAttributes(Math.round(Float.valueOf(nextStation[totalVol])),
nextStation[abNode]);
                station1.addAttributes(Math.round(Float.valueOf(nextStation[diff15_18])),
nextStation[abNode]);
                station1.addAttributes(Math.round(Float.valueOf(nextStation[diff16_19])),
nextStation[abNode]);
            }

            if (Math.round(Float.valueOf(nextStation[funcClass])) == 2){
                station2.addLink(nextStation[abNode]);

station2.addAttributes(Math.round(Float.valueOf(nextStation[count15_18])),
nextStation[abNode]);

station2.addAttributes(Math.round(Float.valueOf(nextStation[count16_19])),
nextStation[abNode]);
                station2.addAttributes(Math.round(Float.valueOf(nextStation[totalVol])),
nextStation[abNode]);
                station2.addAttributes(Math.round(Float.valueOf(nextStation[diff15_18])),
nextStation[abNode]);
                station2.addAttributes(Math.round(Float.valueOf(nextStation[diff16_19])),
nextStation[abNode]);
            }

            if (Math.round(Float.valueOf(nextStation[funcClass])) == 3){
                station3.addLink(nextStation[abNode]);

station3.addAttributes(Math.round(Float.valueOf(nextStation[count15_18])),
nextStation[abNode]);

station3.addAttributes(Math.round(Float.valueOf(nextStation[count16_19])),
nextStation[abNode]);
                station3.addAttributes(Math.round(Float.valueOf(nextStation[totalVol])),
nextStation[abNode]);
                station3.addAttributes(Math.round(Float.valueOf(nextStation[diff15_18])),
nextStation[abNode]);
                station3.addAttributes(Math.round(Float.valueOf(nextStation[diff16_19])),
nextStation[abNode]);
            }

            if (Math.round(Float.valueOf(nextStation[funcClass])) == 4){
                station4.addLink(nextStation[abNode]);

station4.addAttributes(Math.round(Float.valueOf(nextStation[count15_18])),
nextStation[abNode]);
```

```
station4.addAttributes(Math.round(Float.valueOf(nextStation[count16_19])),
nextStation[abNode]);
                station4.addAttributes(Math.round(Float.valueOf(nextStation[totalVol])),
nextStation[abNode]);
                station4.addAttributes(Math.round(Float.valueOf(nextStation[diff15_18])),
nextStation[abNode]);
                station4.addAttributes(Math.round(Float.valueOf(nextStation[diff16_19])),
nextStation[abNode]);
            }

            if (Math.round(Float.valueOf(nextStation[funcClass])) == 5){
                station5.addLink(nextStation[abNode]);

station5.addAttributes(Math.round(Float.valueOf(nextStation[count15_18])),
nextStation[abNode]);

station5.addAttributes(Math.round(Float.valueOf(nextStation[count16_19])),
nextStation[abNode]);
                station5.addAttributes(Math.round(Float.valueOf(nextStation[totalVol])),
nextStation[abNode]);
                station5.addAttributes(Math.round(Float.valueOf(nextStation[diff15_18])),
nextStation[abNode]);
                station5.addAttributes(Math.round(Float.valueOf(nextStation[diff16_19])),
nextStation[abNode]);
            }

            if (Math.round(Float.valueOf(nextStation[funcClass])) == 7){
                station7.addLink(nextStation[abNode]);

station7.addAttributes(Math.round(Float.valueOf(nextStation[count15_18])),
nextStation[abNode]);

station7.addAttributes(Math.round(Float.valueOf(nextStation[count16_19])),
nextStation[abNode]);
                station7.addAttributes(Math.round(Float.valueOf(nextStation[totalVol])),
nextStation[abNode]);
                station7.addAttributes(Math.round(Float.valueOf(nextStation[diff15_18])),
nextStation[abNode]);
                station7.addAttributes(Math.round(Float.valueOf(nextStation[diff16_19])),
nextStation[abNode]);
            }

            if (Math.round(Float.valueOf(nextStation[funcClass])) == 8){
                station8.addLink(nextStation[abNode]);
```

```
station8.addAttributes(Math.round(Float.valueOf(nextStation[count15_18])),
nextStation[abNode]);

station8.addAttributes(Math.round(Float.valueOf(nextStation[count16_19])),
nextStation[abNode]);
                station8.addAttributes(Math.round(Float.valueOf(nextStation[totalVol])),
nextStation[abNode]);
                station8.addAttributes(Math.round(Float.valueOf(nextStation[diff15_18])),
nextStation[abNode]);
                station8.addAttributes(Math.round(Float.valueOf(nextStation[diff16_19])),
nextStation[abNode]);
            }

            station.addLink(nextStation[abNode]);
            station.addAttributes(Math.round(Float.valueOf(nextStation[count15_18])),
nextStation[abNode]);
            station.addAttributes(Math.round(Float.valueOf(nextStation[count16_19])),
nextStation[abNode]);
            station.addAttributes(Math.round(Float.valueOf(nextStation[totalVol])),
nextStation[abNode]);
            station.addAttributes(Math.round(Float.valueOf(nextStation[diff15_18])),
nextStation[abNode]);
            station.addAttributes(Math.round(Float.valueOf(nextStation[diff16_19])),
nextStation[abNode]);

          }
       }catch (Exception e) {
          throw new RuntimeException(e);
       }finally {}

       System.out.println(station.size()+ " total links");
       System.out.println(station1.size()+ " interstate highway links");
       System.out.println(station2.size()+ " freeway&expressway links");
       System.out.println(station3.size() + " major arterial links");
       System.out.println(station4.size() + " minor arterial links");
       System.out.println(station5.size() + " major collector links");
       System.out.println(station7.size() + " local links");
       System.out.println(station8.size() + " high-speed ramp links");

 //All
       System.out.println("Calculating Validation Statistics");
       //Read total volumes and observed counts and store into array lists.
       double[] volume = new double [station.size()];
       DescriptiveStatistics statsVol = new DescriptiveStatistics  ();
       double[] countFive = new double [station.size()];
```

```
double[] countSix = new double [station.size()];
DescriptiveStatistics   diffFive = new DescriptiveStatistics  ();
DescriptiveStatistics   diffSix = new DescriptiveStatistics  ();
DescriptiveStatistics   statsFive = new DescriptiveStatistics  ();
DescriptiveStatistics   statsSix = new DescriptiveStatistics  ();

Iterator<linkRecord> itrVol = station.iterator();
for (int i = 0; i < station.size();i++){
   while (itrVol.hasNext()){
      List<Integer> listVol = itrVol.next().getAttributes();
      if (listVol.get(2) != 0) {  //exclude zero-flow validation links
         volume[i] = (double) listVol.get(2);
         statsVol.addValue((double) listVol.get(2));
         countFive[i] = (double) listVol.get(0);
         countSix[i] = (double) listVol.get(1);
         statsFive.addValue((double) listVol.get(0));
         statsSix.addValue((double) listVol.get(1));
         diffFive.addValue((double) listVol.get(3));
         diffSix.addValue((double) listVol.get(4));
         break;
      }
   }
//System.out.println(volume[i]);
}

System.out.println();
System.out.println("Calculating Squared Pearson Correlation");
double corrFive = new PearsonsCorrelation().correlation(volume, countFive);
double corrSix = new PearsonsCorrelation().correlation(volume, countSix);
double rSquareFive = corrFive*corrFive;
double rSquareSix = corrSix*corrSix;

System.out.println ("R-Square 3pm-5pm = " + rSquareFive);
System.out.println ("R-Square 4pm-6pm = " + rSquareSix);

System.out.println();
System.out.println("Calculating Root Mean Square Error");
double sumSquareDiffFive = diffFive.getSumsq();
double sumSquareDiffSix = diffSix.getSumsq();
double rmseFive = Math.sqrt(sumSquareDiffFive/station.size());
double rmseSix = Math.sqrt(sumSquareDiffSix/station.size());
System.out.println ("Root Mean Square Error 3pm-5pm = " + rmseFive);
System.out.println ("Root Mean Square Error 4pm-6pm = " + rmseSix);

System.out.println();
System.out.println("Calculating Percentage Root Mean Square Error");
```

```
        double pctgRmseFive = rmseFive * 100 * station.size() / statsVol.getSum();
        double pctgRmseSix = rmseSix * 100 * station.size() / statsVol.getSum();
        System.out.println ("% Root Mean Square Error 3pm-5pm = " + pctgRmseFive +
"%");
        System.out.println ("% Root Mean Square Error 4pm-6pm = " + pctgRmseSix +
"%");


        System.out.println("Calculating Absolute Error and Difference of Standard
Deviations");
        double pctFive = 100*(statsVol.getSum()- statsFive.getSum())/statsFive.getSum();
        double pctSix = 100*(statsVol.getSum()- statsSix.getSum())/statsSix.getSum();
        double avgErrorFive = statsVol.getMean() - statsFive.getMean();
        double avgErrorSix = statsVol.getMean() - statsSix.getMean();
        double dsdFive = statsVol.getStandardDeviation() -
statsFive.getStandardDeviation();
        double dsdSix = statsVol.getStandardDeviation() - statsSix.getStandardDeviation();
        System.out.println ("% Diff 3pm-5pm = " + pctFive);
        System.out.println ("% Diff 4pm-6pm = " + pctSix);
        System.out.println ("Average Error 3pm-5pm = " + avgErrorFive);
        System.out.println ("Average Error 4pm-6pm = " + avgErrorSix);
        System.out.println ("DSD 3pm-5pm = " + dsdFive);
        System.out.println ("DSD 4pm-6pm = " + dsdSix);

        try {
           FileWriter fileAll = new FileWriter("resultsAll.csv",true);
           CSVWriter writer = new CSVWriter(fileAll, '\t');
           //write out validation results for all links
           String resultsAll[] =
{String.valueOf(rSquareFive),",",String.valueOf(rSquareSix),",",
                 String.valueOf(rmseFive),",",String.valueOf(rmseSix),",",
                 String.valueOf(pctgRmseFive),",",String.valueOf(pctgRmseSix),",",
                 String.valueOf(pctFive),",",String.valueOf(pctSix),",",
                 String.valueOf(avgErrorFive),",",String.valueOf(avgErrorSix),",",
                 String.valueOf(dsdFive),",",String.valueOf(dsdSix)};
           writer.writeNext(resultsAll, false);
           writer.close();
        }catch (Exception e) {
           throw new RuntimeException(e);
        }finally {}

 //Interstate
        System.out.println();
        System.out.println("Calculating Validation Statistics for Interstate Highway");
        //Read total volumes and observed counts and store into array lists.
        double[] volume1 = new double [station1.size()];
```

```
        DescriptiveStatistics statsVol1 = new DescriptiveStatistics  ();
        double[] countFive1 = new double [station1.size()];
        double[] countSix1 = new double [station1.size()];
        DescriptiveStatistics   diffFive1 = new DescriptiveStatistics  ();
        DescriptiveStatistics   diffSix1 = new DescriptiveStatistics  ();
        DescriptiveStatistics   statsFive1 = new DescriptiveStatistics  ();
        DescriptiveStatistics   statsSix1 = new DescriptiveStatistics  ();

        Iterator<linkRecord> itrVol1 = station1.iterator();
        for (int i = 0; i < station1.size();i++){
           while (itrVol1.hasNext()){
              List<Integer> listVol1 = itrVol1.next().getAttributes();
              if(listVol1.get(2) != 0) {
                 volume1[i] = (double) listVol1.get(2);
                 statsVol1.addValue((double) listVol1.get(2));
                 countFive1[i] = (double) listVol1.get(0);
                 countSix1[i] = (double) listVol1.get(1);
                 statsFive1.addValue((double) listVol1.get(0));
                 statsSix1.addValue((double) listVol1.get(1));
                 diffFive1.addValue((double) listVol1.get(3));
                 diffSix1.addValue((double) listVol1.get(4));
                 break;
              }
           }
           //System.out.println(volume[i]);
        }

        System.out.println();
        System.out.println("Calculating Squared Pearson Correlation");
        double corrFive1 = new PearsonsCorrelation().correlation(volume1, countFive1);
        double corrSix1 = new PearsonsCorrelation().correlation(volume1, countSix1);
        double rSquareFive1 = corrFive1*corrFive1;
        double rSquareSix1 = corrSix1*corrSix1;

        System.out.println ("R-Square 3pm-5pm, Interstate = " + rSquareFive1);
        System.out.println ("R-Square 4pm-6pm, Interstate = " + rSquareSix1);

        System.out.println();
        System.out.println("Calculating Root Mean Square Error");
        double sumSquareDiffFive1 = diffFive1.getSumsq();
        double sumSquareDiffSix1 = diffSix1.getSumsq();
        double rmseFive1 = Math.sqrt(sumSquareDiffFive1/station1.size());
        double rmseSix1 = Math.sqrt(sumSquareDiffSix1/station1.size());
        System.out.println ("Root Mean Square Error 3pm-5pm, Interstate = " +
rmseFive1);
        System.out.println ("Root Mean Square Error 4pm-6pm, Interstate = " + rmseSix1);
```

```
        System.out.println();
        System.out.println("Calculating Percentage Root Mean Square Error");
        double pctgRmseFive1 = rmseFive1 * 100 * station1.size() / statsVol1.getSum();
        double pctgRmseSix1 = rmseSix1 * 100 * station1.size() / statsVol1.getSum();
        System.out.println ("% Root Mean Square Error 3pm-5pm = " + pctgRmseFive1 +
"%");
        System.out.println ("% Root Mean Square Error 4pm-6pm = " + pctgRmseSix1 +
"%");


        System.out.println("Calculating Absolute Error and Difference of Standard
Deviations");
        double pctFive1 = 100*(statsVol1.getSum()-
statsFive1.getSum())/statsFive1.getSum();
        double pctSix1 = 100*(statsVol1.getSum()-
statsSix1.getSum())/statsSix1.getSum();
        double avgErrorFive1 = statsVol1.getMean() - statsFive1.getMean();
        double avgErrorSix1 = statsVol1.getMean() - statsSix1.getMean();
        double dsdFive1 = statsVol1.getStandardDeviation() -
statsFive1.getStandardDeviation();
        double dsdSix1 = statsVol1.getStandardDeviation() -
statsSix1.getStandardDeviation();
        System.out.println ("Average Error 3pm-5pm, Interstate = " + avgErrorFive1);
        System.out.println ("Average Error 4pm-6pm, Interstate = " + avgErrorSix1);
        System.out.println ("DSD 3pm-5pm, Interstate = " + dsdFive1);
        System.out.println ("DSD 4pm-6pm, Interstate = " + dsdSix1);

        try {
           FileWriter fileInt = new FileWriter("resultsInt.csv",true);
           CSVWriter writerInt = new CSVWriter(fileInt, '\t');
           //write out validation results for interstate links
           String resultsInt[] =
{String.valueOf(rSquareFive1),",",String.valueOf(rSquareSix1),",",
                 String.valueOf(rmseFive1),",",String.valueOf(rmseSix1),",",
                 String.valueOf(pctgRmseFive1), ",",String.valueOf(pctgRmseSix1),",",
                 String.valueOf(pctFive1),",",String.valueOf(pctSix1),",",
                 String.valueOf(avgErrorFive1),",",String.valueOf(avgErrorSix1),",",
                 String.valueOf(dsdFive1),",",String.valueOf(dsdSix1)};
           writerInt.writeNext(resultsInt, false);
           writerInt.close();
        }catch (Exception e) {
           throw new RuntimeException(e);
        }finally {}

 //Freeway
```

```
System.out.println();
System.out.println("Calculating Validation Statistics for Freeway & Expressway");
//Read total volumes and observed counts and store into array lists.
double[] volume2 = new double [station2.size()];
DescriptiveStatistics statsVol2 = new DescriptiveStatistics  ();
double[] countFive2 = new double [station2.size()];
double[] countSix2 = new double [station2.size()];
DescriptiveStatistics   diffFive2 = new DescriptiveStatistics  ();
DescriptiveStatistics   diffSix2 = new DescriptiveStatistics  ();
DescriptiveStatistics   statsFive2 = new DescriptiveStatistics  ();
DescriptiveStatistics   statsSix2 = new DescriptiveStatistics  ();

Iterator<linkRecord> itrVol2 = station2.iterator();
for (int i = 0; i < station2.size();i++){
   while (itrVol2.hasNext()){
      List<Integer> listVol2 = itrVol2.next().getAttributes();
      if(listVol2.get(2) != 0) {
         volume2[i] = (double) listVol2.get(2);
         statsVol2.addValue((double) listVol2.get(2));
         countFive2[i] = (double) listVol2.get(0);
         countSix2[i] = (double) listVol2.get(1);
         statsFive2.addValue((double) listVol2.get(0));
         statsSix2.addValue((double) listVol2.get(1));
         diffFive2.addValue((double) listVol2.get(3));
         diffSix2.addValue((double) listVol2.get(4));
         break;
      }
   }
   //System.out.println(volume[i]);
}

System.out.println();
System.out.println("Calculating Squared Pearson Correlation");
double corrFive2 = new PearsonsCorrelation().correlation(volume2, countFive2);
double corrSix2 = new PearsonsCorrelation().correlation(volume2, countSix2);
double rSquareFive2 = corrFive2*corrFive2;
double rSquareSix2 = corrSix2*corrSix2;

System.out.println ("R-Square 3pm-5pm, Interstate = " + rSquareFive2);
System.out.println ("R-Square 4pm-6pm, Interstate = " + rSquareSix2);

System.out.println();
System.out.println("Calculating Root Mean Square Error");
double sumSquareDiffFive2 = diffFive2.getSumsq();
double sumSquareDiffSix2 = diffSix2.getSumsq();
double rmseFive2 = Math.sqrt(sumSquareDiffFive2/station2.size());
```

```
        double rmseSix2 = Math.sqrt(sumSquareDiffSix2/station2.size());
        System.out.println ("Root Mean Square Error 3pm-5pm, Interstate = " +
rmseFive2);
        System.out.println ("Root Mean Square Error 4pm-6pm, Interstate = " + rmseSix2);

        System.out.println();
        System.out.println("Calculating Percentage Root Mean Square Error");
        double pctgRmseFive2 = rmseFive2 * 100 * station2.size() / statsVol2.getSum();
        double pctgRmseSix2 = rmseSix2 * 100 * station2.size() / statsVol2.getSum();
        System.out.println ("% Root Mean Square Error 3pm-5pm = " + pctgRmseFive2 +
"%");
        System.out.println ("% Root Mean Square Error 4pm-6pm = " + pctgRmseSix2 +
"%");

        System.out.println("Calculating Absolute Error and Difference of Standard
Deviations");
        double pctFive2 = 100*(statsVol2.getSum()-
statsFive2.getSum())/statsFive2.getSum();
        double pctSix2 = 100*(statsVol2.getSum()-
statsSix2.getSum())/statsSix2.getSum();
        double avgErrorFive2 = statsVol2.getMean() - statsFive2.getMean();
        double avgErrorSix2 = statsVol2.getMean() - statsSix2.getMean();
        double dsdFive2 = statsVol2.getStandardDeviation() -
statsFive2.getStandardDeviation();
        double dsdSix2 = statsVol2.getStandardDeviation() -
statsSix2.getStandardDeviation();
        System.out.println ("Average Error 3pm-5pm, Interstate = " + avgErrorFive2);
        System.out.println ("Average Error 4pm-6pm, Interstate = " + avgErrorSix2);
        System.out.println ("DSD 3pm-5pm, Interstate = " + dsdFive2);
        System.out.println ("DSD 4pm-6pm, Interstate = " + dsdSix2);

        try {
          FileWriter fileFree = new FileWriter("resultsFree.csv",true);
          CSVWriter writerFree = new CSVWriter(fileFree, '\t');
          //write out validation results for free-way links
          String resultsFree[] =
{String.valueOf(rSquareFive2),",",String.valueOf(rSquareSix2),",",
                String.valueOf(rmseFive2),",",String.valueOf(rmseSix2),",",
                String.valueOf(pctgRmseFive2), ",",String.valueOf(pctgRmseSix2),",",
                String.valueOf(pctFive2),",",String.valueOf(pctSix2),",",
                String.valueOf(avgErrorFive2),",",String.valueOf(avgErrorSix2),",",
                String.valueOf(dsdFive2),",",String.valueOf(dsdSix2)};
          writerFree.writeNext(resultsFree, false);
          writerFree.close();
        }catch (Exception e) {
          throw new RuntimeException(e);
```

```
        }finally {}

    //Major Arterial
        System.out.println();
        System.out.println("Calculating Validation Statistics for Major Arterial");
        //Read total volumes and observed counts and store into array lists.
        double[] volume3 = new double [station3.size()];
        DescriptiveStatistics statsVol3 = new DescriptiveStatistics  ();
        double[] countFive3 = new double [station3.size()];
        double[] countSix3 = new double [station3.size()];
        DescriptiveStatistics   diffFive3 = new DescriptiveStatistics  ();
        DescriptiveStatistics   diffSix3 = new DescriptiveStatistics  ();
        DescriptiveStatistics   statsFive3 = new DescriptiveStatistics  ();
        DescriptiveStatistics   statsSix3 = new DescriptiveStatistics  ();

        Iterator<linkRecord> itrVol3 = station3.iterator();
        for (int i = 0; i < station3.size();i++){
           while (itrVol3.hasNext()){
              List<Integer> listVol3 = itrVol3.next().getAttributes();
              if(listVol3.get(2) != 0) {
                 volume3[i] = (double) listVol3.get(2);
                 statsVol3.addValue((double) listVol3.get(2));
                 countFive3[i] = (double) listVol3.get(0);
                 countSix3[i] = (double) listVol3.get(1);
                 statsFive3.addValue((double) listVol3.get(0));
                 statsSix3.addValue((double) listVol3.get(1));
                 diffFive3.addValue((double) listVol3.get(3));
                 diffSix3.addValue((double) listVol3.get(4));
                 break;
              }
           }
           //System.out.println(volume[i]);
        }

        System.out.println();
        System.out.println("Calculating Squared Pearson Correlation");
        double corrFive3 = new PearsonsCorrelation().correlation(volume3, countFive3);
        double corrSix3 = new PearsonsCorrelation().correlation(volume3, countSix3);
        double rSquareFive3 = corrFive3*corrFive3;
        double rSquareSix3 = corrSix3*corrSix3;

        System.out.println ("R-Square 3pm-5pm, Interstate = " + rSquareFive3);
        System.out.println ("R-Square 4pm-6pm, Interstate = " + rSquareSix3);

        System.out.println();
        System.out.println("Calculating Root Mean Square Error");
```

```
        double sumSquareDiffFive3 = diffFive3.getSumsq();
        double sumSquareDiffSix3 = diffSix3.getSumsq();
        double rmseFive3 = Math.sqrt(sumSquareDiffFive3/station3.size());
        double rmseSix3 = Math.sqrt(sumSquareDiffSix3/station3.size());
        System.out.println ("Root Mean Square Error 3pm-5pm, Interstate = " +
rmseFive3);
        System.out.println ("Root Mean Square Error 4pm-6pm, Interstate = " + rmseSix3);

        System.out.println();
        System.out.println("Calculating Percentage Root Mean Square Error");
        double pctgRmseFive3 = rmseFive3 * 100 * station3.size() / statsVol3.getSum();
        double pctgRmseSix3 = rmseSix3 * 100 * station3.size() / statsVol3.getSum();
        System.out.println ("% Root Mean Square Error 3pm-5pm = " + pctgRmseFive3 +
"%");
        System.out.println ("% Root Mean Square Error 4pm-6pm = " + pctgRmseSix3 +
"%");

        System.out.println("Calculating Absolute Error and Difference of Standard
Deviations");
        double pctFive3 = 100*(statsVol3.getSum()-
statsFive3.getSum())/statsFive3.getSum();
        double pctSix3 = 100*(statsVol3.getSum()-
statsSix3.getSum())/statsSix3.getSum();
        double avgErrorFive3 = statsVol3.getMean() - statsFive3.getMean();
        double avgErrorSix3 = statsVol3.getMean() - statsSix3.getMean();
        double dsdFive3 = statsVol3.getStandardDeviation() -
statsFive3.getStandardDeviation();
        double dsdSix3 = statsVol3.getStandardDeviation() -
statsSix3.getStandardDeviation();
        System.out.println ("Average Error 3pm-5pm, Interstate = " + avgErrorFive3);
        System.out.println ("Average Error 4pm-6pm, Interstate = " + avgErrorSix3);
        System.out.println ("DSD 3pm-5pm, Interstate = " + dsdFive3);
        System.out.println ("DSD 4pm-6pm, Interstate = " + dsdSix3);

        try {
           FileWriter fileMajor = new FileWriter("resultsMajor.csv",true);
           CSVWriter writerMajor = new CSVWriter(fileMajor, '\t');
           //write out validation results for major arterial links
           String resultsMajor[] =
{String.valueOf(rSquareFive3),",",String.valueOf(rSquareSix3),",",
                 String.valueOf(rmseFive3),",",String.valueOf(rmseSix3),",",
                 String.valueOf(pctgRmseFive3), ",",String.valueOf(pctgRmseSix3),",",
                 String.valueOf(pctFive3),",",String.valueOf(pctSix3),",",
                 String.valueOf(avgErrorFive3),",",String.valueOf(avgErrorSix3),",",
                 String.valueOf(dsdFive3),",",String.valueOf(dsdSix3)};
           writerMajor.writeNext(resultsMajor, false);
```

```
            writerMajor.close();
        }catch (Exception e) {
            throw new RuntimeException(e);
        }finally {}

//Minor Arterial
        System.out.println();
        System.out.println("Calculating Validation Statistics for Minor Arterial");
        //Read total volumes and observed counts and store into array lists.
        double[] volume4 = new double [station4.size()];
        DescriptiveStatistics statsVol4 = new DescriptiveStatistics  ();
        double[] countFive4= new double [station4.size()];
        double[] countSix4 = new double [station4.size()];
        DescriptiveStatistics   diffFive4 = new DescriptiveStatistics  ();
        DescriptiveStatistics   diffSix4 = new DescriptiveStatistics  ();
        DescriptiveStatistics   statsFive4 = new DescriptiveStatistics  ();
        DescriptiveStatistics   statsSix4 = new DescriptiveStatistics  ();

        Iterator<linkRecord> itrVol4 = station4.iterator();
        for (int i = 0; i < station4.size();i++){
            while (itrVol4.hasNext()){
                List<Integer> listVol4 = itrVol4.next().getAttributes();
                if(listVol4.get(2) != 0) {
                    volume4[i] = (double) listVol4.get(2);
                    statsVol4.addValue((double) listVol4.get(2));
                    countFive4[i] = (double) listVol4.get(0);
                    countSix4[i] = (double) listVol4.get(1);
                    statsFive4.addValue((double) listVol4.get(0));
                    statsSix4.addValue((double) listVol4.get(1));
                    diffFive4.addValue((double) listVol4.get(3));
                    diffSix4.addValue((double) listVol4.get(4));
                    break;
                }
            }
            //System.out.println(volume[i]);
        }

        System.out.println();
        System.out.println("Calculating Squared Pearson Correlation");
        double corrFive4 = new PearsonsCorrelation().correlation(volume4, countFive4);
        double corrSix4 = new PearsonsCorrelation().correlation(volume4, countSix4);
        double rSquareFive4 = corrFive4*corrFive4;
        double rSquareSix4 = corrSix4*corrSix4;

        System.out.println ("R-Square 3pm-5pm, Interstate = " + rSquareFive4);
        System.out.println ("R-Square 4pm-6pm, Interstate = " + rSquareSix4);
```

```
        System.out.println();
        System.out.println("Calculating Root Mean Square Error");
        double sumSquareDiffFive4 = diffFive4.getSumsq();
        double sumSquareDiffSix4 = diffSix4.getSumsq();
        double rmseFive4 = Math.sqrt(sumSquareDiffFive4/station4.size());
        double rmseSix4 = Math.sqrt(sumSquareDiffSix4/station4.size());
        System.out.println ("Root Mean Square Error 3pm-5pm, Interstate = " +
rmseFive4);
        System.out.println ("Root Mean Square Error 4pm-6pm, Interstate = " + rmseSix4);

        System.out.println();
        System.out.println("Calculating Percentage Root Mean Square Error");
        double pctgRmseFive4 = rmseFive4 * 100 * station4.size() / statsVol4.getSum();
        double pctgRmseSix4 = rmseSix4 * 100 * station4.size() / statsVol4.getSum();
        System.out.println ("% Root Mean Square Error 3pm-5pm = " + pctgRmseFive4 +
"%");
        System.out.println ("% Root Mean Square Error 4pm-6pm = " + pctgRmseSix4 +
"%");

        System.out.println("Calculating Absolute Error and Difference of Standard
Deviations");
        double pctFive4 = 100*(statsVol4.getSum()-
statsFive4.getSum())/statsFive4.getSum();
        double pctSix4 = 100*(statsVol4.getSum()-
statsSix4.getSum())/statsSix4.getSum();
        double avgErrorFive4 = statsVol4.getMean() - statsFive4.getMean();
        double avgErrorSix4 = statsVol4.getMean() - statsSix4.getMean();
        double dsdFive4 = statsVol4.getStandardDeviation() -
statsFive4.getStandardDeviation();
        double dsdSix4 = statsVol4.getStandardDeviation() -
statsSix4.getStandardDeviation();
        System.out.println ("Average Error 3pm-5pm, Interstate = " + avgErrorFive4);
        System.out.println ("Average Error 4pm-6pm, Interstate = " + avgErrorSix4);
        System.out.println ("DSD 3pm-5pm, Interstate = " + dsdFive4);
        System.out.println ("DSD 4pm-6pm, Interstate = " + dsdSix4);

        try {
           FileWriter fileMinor = new FileWriter("resultsMinor.csv",true);
           CSVWriter writerMinor = new CSVWriter(fileMinor, '\t');
           //write out validation results for minor arterial links
           String resultsMinor[] =
{String.valueOf(rSquareFive4),",",String.valueOf(rSquareSix4),",",
                 String.valueOf(rmseFive4),",",String.valueOf(rmseSix4),",",
                 String.valueOf(pctgRmseFive4), ",",String.valueOf(pctgRmseSix4),",",
                 String.valueOf(pctFive4),",",String.valueOf(pctSix4),",",
```

```
                String.valueOf(avgErrorFive4),",",String.valueOf(avgErrorSix4),",",
                String.valueOf(dsdFive4),",",String.valueOf(dsdSix4)};
            writerMinor.writeNext(resultsMinor, false);
            writerMinor.close();
        }catch (Exception e) {
            throw new RuntimeException(e);
        }finally {}

//Collector
        System.out.println();
        System.out.println("Calculating Validation Statistics for Major Collector");
        //Read total volumes and observed counts and store into array lists.
        double[] volume5 = new double [station5.size()];
        DescriptiveStatistics statsVol5 = new DescriptiveStatistics  ();
        double[] countFive5= new double [station5.size()];
        double[] countSix5 = new double [station5.size()];
        DescriptiveStatistics   diffFive5 = new DescriptiveStatistics  ();
        DescriptiveStatistics   diffSix5 = new DescriptiveStatistics  ();
        DescriptiveStatistics   statsFive5 = new DescriptiveStatistics  ();
        DescriptiveStatistics   statsSix5 = new DescriptiveStatistics  ();

        Iterator<linkRecord> itrVol5 = station5.iterator();
        for (int i = 0; i < station5.size();i++){
            while (itrVol5.hasNext()){
                List<Integer> listVol5 = itrVol5.next().getAttributes();
                if(listVol5.get(2) != 0) {
                    volume5[i] = (double) listVol5.get(2);
                    statsVol5.addValue((double) listVol5.get(2));
                    countFive5[i] = (double) listVol5.get(0);
                    countSix5[i] = (double) listVol5.get(1);
                    statsFive5.addValue((double) listVol5.get(0));
                    statsSix5.addValue((double) listVol5.get(1));
                    diffFive5.addValue((double) listVol5.get(3));
                    diffSix5.addValue((double) listVol5.get(4));
                    break;
                }
            }
            //System.out.println(volume[i]);
        }

        System.out.println();
        System.out.println("Calculating Squared Pearson Correlation");
        double corrFive5 = new PearsonsCorrelation().correlation(volume5, countFive5);
        double corrSix5 = new PearsonsCorrelation().correlation(volume5, countSix5);
        double rSquareFive5 = corrFive5*corrFive5;
        double rSquareSix5 = corrSix5*corrSix5;
```

```
        System.out.println ("R-Square 3pm-5pm, Interstate = " + rSquareFive5);
        System.out.println ("R-Square 4pm-6pm, Interstate = " + rSquareSix5);

        System.out.println();
        System.out.println("Calculating Root Mean Square Error");
        double sumSquareDiffFive5 = diffFive5.getSumsq();
        double sumSquareDiffSix5 = diffSix5.getSumsq();
        double rmseFive5 = Math.sqrt(sumSquareDiffFive5/station5.size());
        double rmseSix5 = Math.sqrt(sumSquareDiffSix5/station5.size());
        System.out.println ("Root Mean Square Error 3pm-5pm, Interstate = " +
rmseFive5);
        System.out.println ("Root Mean Square Error 4pm-6pm, Interstate = " + rmseSix5);

        System.out.println();
        System.out.println("Calculating Percentage Root Mean Square Error");
        double pctgRmseFive5 = rmseFive5 * 100 * station5.size() / statsVol5.getSum();
        double pctgRmseSix5 = rmseSix5 * 100 * station5.size() / statsVol5.getSum();
        System.out.println ("% Root Mean Square Error 3pm-5pm = " + pctgRmseFive5 +
"%");
        System.out.println ("% Root Mean Square Error 4pm-6pm = " + pctgRmseSix5 +
"%");

        System.out.println("Calculating Absolute Error and Difference of Standard
Deviations");
        double pctFive5 = 100*(statsVol5.getSum()-
statsFive5.getSum())/statsFive5.getSum();
        double pctSix5 = 100*(statsVol5.getSum()-
statsSix5.getSum())/statsSix5.getSum();
        double avgErrorFive5 = statsVol5.getMean() - statsFive5.getMean();
        double avgErrorSix5 = statsVol5.getMean() - statsSix5.getMean();
        double dsdFive5 = statsVol5.getStandardDeviation() -
statsFive5.getStandardDeviation();
        double dsdSix5 = statsVol5.getStandardDeviation() -
statsSix5.getStandardDeviation();
        System.out.println ("Average Error 3pm-5pm, Interstate = " + avgErrorFive5);
        System.out.println ("Average Error 4pm-6pm, Interstate = " + avgErrorSix5);
        System.out.println ("DSD 3pm-5pm, Interstate = " + dsdFive5);
        System.out.println ("DSD 4pm-6pm, Interstate = " + dsdSix5);

        try {
           FileWriter fileCol = new FileWriter("resultsCol.csv",true);
           CSVWriter writerCol = new CSVWriter(fileCol, '\t');
           //write out validation results for collector links
           String resultsCol[] =
{String.valueOf(rSquareFive5),",",String.valueOf(rSquareSix5),",",
```

```
                String.valueOf(rmseFive5),",",String.valueOf(rmseSix5),",",
                String.valueOf(pctgRmseFive5), ",",String.valueOf(pctgRmseSix5),",",
                String.valueOf(pctFive5),",",String.valueOf(pctSix5),",",
                String.valueOf(avgErrorFive5),",",String.valueOf(avgErrorSix5),",",
                String.valueOf(dsdFive5),",",String.valueOf(dsdSix5)};
            writerCol.writeNext(resultsCol, false);
            writerCol.close();
        }catch (Exception e) {
            throw new RuntimeException(e);
        }finally {}
    }

}
```

Cube Script – linkIndex.s

```
; Yuchen's Dissertation
; This script reads Dissert_Veh_PM.net and index its links
; Main Directory: Models\NetworkDefiningModel\Analysis
;Yuchen Cui - 07/25/2015

 v = '5'

 RUN PGM=NETWORK  MSG = 'Convert output NET to DBF'
    NETI = Assignment\Dissert_Veh_PM@v@.net
    LINKO = TraceOD\81zones\Dissert_Veh_PM@v@_link.dbf
 ENDRUN

 RUN PGM=MATRIX MSG = 'Index link and convert DBF to CSV'

 FILEI DBI[1] = TraceOD\81zones\Dissert_Veh_PM@v@_link.dbf
 FILEO PRINTO[1] = TraceOD\81zones\linkIndex_@v@.csv

 ZONES = 1

 LOOP _K = 1, DBI.1.NUMRECORDS
 ;loop over all record in the DBF file
     _myRec = DBIREADRECORD(1,_K)
     ;read first record in DBF file
     myAB = Ltrim(Str(DI.1.A,10,0))+ '_' + Ltrim(Str(DI.1.B,10,0))

     PRINT LIST = myAB(T), DI.1.A(N), DI.1.B(N), DI.1.TOTAL_VOL(N),
DI.1.VolCap, _K PRINTO=1
                        CSV=T
     ;after record is read, use DI.#.fieldName to retrive the value
 ENDLOOP
```

```
ENDRUN
```

Cube Script – removeLinks.s

```
; Yuchen's Dissertation
; This script remove low trip-share links in baltNetwork_run.net
; Main Directory: Models\NetworkDefiningModel\Analysis
;Yuchen Cui - 10/29/2015

v = '5', v1 = '6'

RUN PGM=NETWORK  MSG = 'Convert output NET to DBF'
   NETI = Assignment\baltNetwork_run@v@.net
   LINKO = NetworkFile\baltNetwork_run@v@_link.dbf
   NODEO = NetworkFile\baltNetwork_run@v@_node.dbf
ENDRUN

RUN PGM=MATRIX MSG = 'remove low trip-share in baltNetwork_run.net'

FILEO RECO[1] = NetworkFile\baltNetwork_run@v@_updateLink.dbf,
FIELDS= A, B, ROUTEID, ROADNAMELO, ROADNAMESH, ID_PREFIX,
ID_RTE_NO, FIPS,
        TOLL_AM, TOLL_PM, BMP, EMP, SMZRMZ, AREATYPE,
        LANE, FUNCCLASS, FFSPD, CAPACITY, FUNCTIONAL, DISTANCE,
ONEWAY,
        PMLIMIT, A_B, DISABLED
FILEI DBI[2] = NetworkFile\baltNetwork_run@v@_link.dbf
;this file contains full link information of the entire network
;IMPORTANT: in order to use DBISeek, users need to sort all arguments to
;be used in DBISeek in the script later

FILEI DBI[1] = TraceOD\81zones\excLink_@v@.csv, A=1,B=2, SORT = A, B
;this file is provides the links to be extracted
;CSV file can be inputed as DBI, and A(N)=1 indicates the 1st data field's
;name and B(N)=2 indicates the 2nd data field's name

ZONES = 1

LOOP  _K = 1, DBI.2.NUMRECORDS
;for each link to be extracted. We skip the first row since it's the header.
     _READ1 = DBIREADRECORD(2, _K)
    ;get A node and B node
     myA = DI.2.A
     myB = DI.2.B
     myROUTEID = DI.2.ROUTEID
     myROADNAMELO = DI.2.ROADNAMELO
     myROADNAMESH = DI.2.ROADNAMESH
```

```
myID_PREFIX = DI.2.ID_PREFIX
myID_RTE_NO = DI.2.ID_RTE_NO
myFIPS = DI.2.FIPS
myTOLL_AM = DI.2.TOLL_AM
myTOLL_PM = DI.2.TOLL_PM
myBMP = DI.2.BMP
myEMP = DI.2.EMP
mySMZRMZ = DI.2.SMZRMZ
myAREATYPE = DI.2.AREATYPE
myLANE = DI.2.LANE
myFUNCCLASS = DI.2.FUNCCLASS
myFFSPD = DI.2.FFSPD
myCAPACITY = DI.2.CAPACITY
myFUNCTIONAL = DI.2.FUNCTIONAL
myDISTANCE = DI.2.DISTANCE
myONEWAY = DI.2.ONEWAY
myPMLIMIT = DI.2.PMLIMIT
myAB = Ltrim(Str(DI.2.A,10,0))+ '_' + Ltrim(Str(DI.2.B,10,0))
myDISABLED = DI.2.DISABLED

IF (DBISEEK(1, myA, myB) == 0)
;find the specific link record index in the CSV link file
   RO.A = myA
   ;if found, update DISABLED field to 1
   RO.B = myB
   RO.ROUTEID = myROUTEID
   RO.ROADNAMELO = myROADNAMELO
   RO.ROADNAMESH = myROADNAMESH
   RO.ID_PREFIX = myID_PREFIX
   RO.ID_RTE_NO = myID_RTE_NO
   RO.FIPS = myFIPS
   RO.TOLL_AM = myTOLL_AM
   RO.TOLL_PM = myTOLL_PM
   RO.BMP = myBMP
   RO.EMP = myEMP
   RO.SMZRMZ = mySMZRMZ
   RO.AREATYPE = myAREATYPE
   RO.LANE = myLANE
   RO.FUNCCLASS = myFUNCCLASS
   RO.FFSPD = myFFSPD
   RO.CAPACITY = myCAPACITY
   RO.FUNCTIONAL = myFUNCTIONAL
   RO.DISTANCE = myDISTANCE
   RO.ONEWAY = myONEWAY
   RO.PMLIMIT = myPMLIMIT
   RO.A_B = myAB
```

```
        RO.DISABLED = 1
    ELSE
        RO.A = myA
        ;if not found, just write out original attributes, where DISABLED is 0
        RO.B = myB
        RO.ROUTEID = myROUTEID
        RO.ROADNAMELO = myROADNAMELO
        RO.ROADNAMESH = myROADNAMESH
        RO.ID_PREFIX = myID_PREFIX
        RO.ID_RTE_NO = myID_RTE_NO
        RO.FIPS = myFIPS
        RO.TOLL_AM = myTOLL_AM
        RO.TOLL_PM = myTOLL_PM
        RO.BMP = myBMP
        RO.EMP = myEMP
        RO.SMZRMZ = mySMZRMZ
        RO.AREATYPE = myAREATYPE
        RO.LANE = myLANE
        RO.FUNCCLASS = myFUNCCLASS
        RO.FFSPD = myFFSPD
        RO.CAPACITY = myCAPACITY
        RO.FUNCTIONAL = myFUNCTIONAL
        RO.DISTANCE = myDISTANCE
        RO.ONEWAY = myONEWAY
        RO.PMLIMIT = myPMLIMIT
        RO.A_B = myAB
        RO.DISABLED = myDISABLED
    ENDIF

    WRITE RECO = 1
    ;write out the record
ENDLOOP
ENDRUN

RUN PGM=NETWORK  MSG='Network - Convert to DBF-2 to NETWORK'
LINKI = NetworkFile\baltNetwork_run@v@_updateLink.dbf
NODEI = NetworkFile\baltNetwork_run@v@_node.dbf

NETO = NetworkFile\baltNetwork_run@v@Disabled.net
ENDRUN

RUN PGM=NETWORK MSG = 'Delete zero-volume links in baltNetwork_run8.net'

FILEO NETO = Assignment\baltNetwork_run@v1@.net
FILEI LINKI[1] = NetworkFile\baltNetwork_run@v@Disabled.net
```

```
 PROCESS  PHASE=LINKMERGE
  IF (LI.1.DISABLED==1)
  delete
  ENDIF

 ENDPROCESS

 ENDRUN
```

Cube Script – exportLinkOD.s

```
; Yuchen's Dissertation
 ; This script export non-zero selTotal values from *.NET to *.DBF;
 ; Main Directory: Models\NetworkDefiningModel
 ; Post processing after OD-split Highway Assignment

v = 5

LOOP _K = 1,2000
RUN PGM=NETWORK PRNFILE='Print\Convert Network to DBF.PRN'
MSG='Export link per OD - Convert Network to DBF-2'
FILEI LINKI[1] = Assignment\81zones\Dissert_Veh_PM@v@_OD@_K@.net
;pay attention to the file name of assignment result
FILEO LINKO = TraceOD\81zones\Dissert_Veh_PM@v@_OD@_K@.dbf

PROCESS  PHASE=LINKMERGE
  IF (LI.1.selTotal==0)
    delete
  ENDIF
ENDPROCESS
ENDRUN

RUN PGM=MATRIX MSG = 'DBF to CSV'
  FILEI DBI[1] = TraceOD\81zones\Dissert_Veh_PM@v@_OD@_K@.dbf
  FILEO PRINTO[1] = TraceOD\81zones\Links@v@_OD@_K@.csv

  ZONES = 1

  LOOP _L = 1, DBI.1.NUMRECORDS
  ;loop over all record in the DBF file
     _myRec = DBIREADRECORD(1,_L)
      ;read first record in DBF file
     _myAB = Ltrim(Str(DI.1.A,10,0))+'_'+Ltrim(Str(DI.1.B,10,0))
     PRINT LIST = _myAB, DI.1.A(N), DI.1.B(N), DI.1.SELTOTAL(N), PRINTO=1
CSV=T
      ;after record is read, use DI.#.fieldName to retrive the value
  ENDLOOP
```

```
ENDRUN

ENDLOOP
```

Cube Script – removeZeroLinks.s

```
; Yuchen's Dissertation
; This script remove zero-volume links in baltNetwork_run.net
; Main Directory: Models\NetworkDefiningModel
;Yuchen Cui - 10/29/2015

 v = '6', v1 = '7'

 RUN PGM=NETWORK  MSG = 'Convert output NET to DBF'
    NETI = Assignment\baltNetwork_run@v@.net
    LINKO = NetworkFile\baltNetwork_run@v@_link.dbf
    NODEO = NetworkFile\baltNetwork_run@v@_node.dbf
 ENDRUN

 RUN PGM=NETWORK  MSG = 'Convert output NET to DBF'
    NETI = Assignment\Dissert_Veh_PM@v@.net
    LINKO = NetworkFile\Dissert_Veh_PM@v@_link.dbf
 ENDRUN

 RUN PGM=MATRIX MSG = 'Convert Dissert_Veh_PM.dbf to zeroLinks.csv'

 FILEI DBI[1] = NetworkFile\Dissert_Veh_PM@v@_link.dbf
 FILEO PRINTO[1] = NetworkFile\zeroLinksPM@v@_81zone.csv

 ZONES = 1

 LOOP _K = 1, DBI.1.NUMRECORDS
 ;loop over all record in the DBF file
     _myRec = DBIREADRECORD(1,_K)
     ;read first record in DBF file
     IF (DI.1.TOTAL_VOL = 0)
        PRINT LIST = DI.1.A(N), DI.1.B(N) PRINTO=1 CSV=T
        ;after record is read, use DI.#.fieldName to retrive the value
     ENDIF
 ENDLOOP

 ENDRUN


 RUN PGM=MATRIX MSG = 'remove low trip-share in baltNetwork_run.net'

 FILEO RECO[1] = NetworkFile\baltNetwork_run@v@_updateLink.dbf,
```

```
 FIELDS= A, B, ROUTEID, ROADNAMELO, ROADNAMESH, ID_PREFIX,
ID_RTE_NO, FIPS,
        TOLL_AM, TOLL_PM, BMP, EMP, SMZRMZ, AREATYPE,
        LANE, FUNCCLASS, FFSPD, CAPACITY, FUNCTIONAL, DISTANCE,
ONEWAY,
        PMLIMIT, A_B, DISABLED

 FILEI DBI[2] = NetworkFile\baltNetwork_run@v@_link.dbf
 ;this file contains full link information of the entire network
 ;IMPORTANT: in order to use DBISeek, users need to sort all arguments to
 ;be used in DBISeek in the script later

 FILEI DBI[1] = NetworkFile\zeroLinksPM@v@_81zone.csv, A=1,B=2, SORT = A, B
 ;this file is provides the links to be extracted
 ;CSV file can be inputed as DBI, and A(N)=1 indicates the 1st data field's
 ;name and B(N)=2 indicates the 2nd data field's name

 ZONES = 1

 LOOP  _K = 1, DBI.2.NUMRECORDS
 ;for each link to be extracted. We skip the first row since it's the header.
      _READ1 = DBIREADRECORD(2, _K)
     ;get A node and B node
      myA = DI.2.A
      myB = DI.2.B
      myROUTEID = DI.2.ROUTEID
      myROADNAMELO = DI.2.ROADNAMELO
      myROADNAMESH = DI.2.ROADNAMESH
      myID_PREFIX = DI.2.ID_PREFIX
      myID_RTE_NO = DI.2.ID_RTE_NO
      myFIPS = DI.2.FIPS
      myTOLL_AM = DI.2.TOLL_AM
      myTOLL_PM = DI.2.TOLL_PM
      myBMP = DI.2.BMP
      myEMP = DI.2.EMP
      mySMZRMZ = DI.2.SMZRMZ
      myAREATYPE = DI.2.AREATYPE
      myLANE = DI.2.LANE
      myFUNCCLASS = DI.2.FUNCCLASS
      myFFSPD = DI.2.FFSPD
      myCAPACITY = DI.2.CAPACITY
      myFUNCTIONAL = DI.2.FUNCTIONAL
      myDISTANCE = DI.2.DISTANCE
      myONEWAY = DI.2.ONEWAY
      myPMLIMIT = DI.2.PMLIMIT
      myAB = Ltrim(Str(DI.2.A,10,0))+ '_' + Ltrim(Str(DI.2.B,10,0))
```

```
 myDISABLED = DI.2.DISABLED

 IF (DBISEEK(1, myA, myB) == 0)
 ;find the specific link record index in the CSV link file
    RO.A = myA
    ;if found, update DISABLED field to 1
    RO.B = myB
    RO.ROUTEID = myROUTEID
    RO.ROADNAMELO = myROADNAMELO
    RO.ROADNAMESH = myROADNAMESH
    RO.ID_PREFIX = myID_PREFIX
    RO.ID_RTE_NO = myID_RTE_NO
    RO.FIPS = myFIPS
    RO.TOLL_AM = myTOLL_AM
    RO.TOLL_PM = myTOLL_PM
    RO.BMP = myBMP
    RO.EMP = myEMP
    RO.SMZRMZ = mySMZRMZ
    RO.AREATYPE = myAREATYPE
    RO.LANE = myLANE
    RO.FUNCCLASS = myFUNCCLASS
    RO.FFSPD = myFFSPD
    RO.CAPACITY = myCAPACITY
    RO.FUNCTIONAL = myFUNCTIONAL
    RO.DISTANCE = myDISTANCE
    RO.ONEWAY = myONEWAY
    RO.PMLIMIT = myPMLIMIT
    RO.A_B = myAB
    RO.DISABLED = 1
ELSE
    RO.A = myA
    ;if not found, just write out original attributes, where DISABLED is 0
    RO.B = myB
    RO.ROUTEID = myROUTEID
    RO.ROADNAMELO = myROADNAMELO
    RO.ROADNAMESH = myROADNAMESH
    RO.ID_PREFIX = myID_PREFIX
    RO.ID_RTE_NO = myID_RTE_NO
    RO.FIPS = myFIPS
    RO.TOLL_AM = myTOLL_AM
    RO.TOLL_PM = myTOLL_PM
    RO.BMP = myBMP
    RO.EMP = myEMP
    RO.SMZRMZ = mySMZRMZ
    RO.AREATYPE = myAREATYPE
    RO.LANE = myLANE
```

```
        RO.FUNCCLASS = myFUNCCLASS
        RO.FFSPD = myFFSPD
        RO.CAPACITY = myCAPACITY
        RO.FUNCTIONAL = myFUNCTIONAL
        RO.DISTANCE = myDISTANCE
        RO.ONEWAY = myONEWAY
        RO.PMLIMIT = myPMLIMIT
        RO.A_B = myAB
        RO.DISABLED = myDISABLED
    ENDIF

    WRITE RECO = 1
    ;write out the record

ENDLOOP
ENDRU

RUN PGM=NETWORK  MSG='Network - Convert to DBF-2 to NETWORK'

LINKI = NetworkFile\baltNetwork_run@v@_updateLink.dbf
NODEI = NetworkFile\baltNetwork_run@v@_node.dbf

NETO = NetworkFile\baltNetwork_run@v@Disabled.net
ENDRUN

RUN PGM=NETWORK MSG = 'Delete zero-volume links in baltNetwork_run8.net'

FILEO NETO = Assignment\baltNetwork_run@v1@.net
FILEI LINKI[1] = NetworkFile\baltNetwork_run@v@Disabled.net

PROCESS  PHASE=LINKMERGE
 IF (LI.1.DISABLED==1)
 delete
 ENDIF

ENDPROCESS
ENDRUN
```

Java Script – sortOD.java

```
/**
 * Created by Yuchen Cui on 6/17/2015.
 * The sortOD application class is used to sort and index OD trip records
 */

import com.opencsv.CSVReader;
import com.opencsv.CSVWriter;
```

```
import com.pb.common.util.IndexSort;

import java.io.*;
import java.util.Iterator;


public class sortOD {
   public static void main(String[] args) throws IOException {
      //define input file names
      String file = "subareaVehPMOD.csv";
      linkDatabase tripRec = new linkDatabase();   //for all OD trip records

      //column numbers in subareaVehPMOD.csv
      int origin = 0;
      int destination = 1;
      int trips = 2;

      System.out.println("Read link file: subareaVehPMOD.csv");
      int seq = 0;
      try {
         CSVReader tripReader = new CSVReader(new FileReader(file));
         String [] nextTrip; //number of OD trips in the table: 106212

         while ((nextTrip = tripReader.readNext()) != null) {
            String tripOrigin = nextTrip[origin];
            String tripDestination = nextTrip[destination];
            String id = String.valueOf(seq);

            //add OD as id to each trip record
            if (!nextTrip[origin].equals(nextTrip[destination])) {
               tripRec.addLink(id);
               //add attributes: origin, destination, trips
               tripRec.addAttributes(Integer.valueOf(nextTrip[origin]), id);
               tripRec.addAttributes(Integer.valueOf(nextTrip[destination]), id);
               tripRec.addAttributes(Math.round(Float.valueOf(nextTrip[trips])), id);


               //System.out.println(tripRec.getAttributes(id).get(trips));
               seq++;
            }
         }
      }catch (Exception e) {
         throw new RuntimeException(e);
      }finally {}

      System.out.println("Write indexed OD records: indexedVehPMOD.csv");
```

```
        int[] trip = new int [tripRec.size()];
        try {
           CSVWriter writer = new CSVWriter(new FileWriter("indexedVehPMOD.csv"),
'\t');
           Iterator<linkRecord> itrCount = tripRec.iterator();

           while (itrCount.hasNext()){
              linkRecord odRec = itrCount.next();
              String tripIndex = odRec.getNum();
              trip[Integer.valueOf(tripIndex)] = odRec.getAttributes().get(2);

              String[] entries = odRec.writeToStrings();
              writer.writeNext(entries, false);
           }
           writer.close();

        }catch (Exception e) {
           throw new RuntimeException(e);
        }finally {}



        System.out.println("Starting to sort trip[]");
        IndexSort sort = new IndexSort();
        int[] tripSortedIndex = sort.indexSort(trip);

        System.out.println("Write sorted trip index to database");
        linkDatabase indexRec = new linkDatabase();

        //print out in ascending order
        for (int i = 0; i < tripRec.size(); i++){
           String indexID = String.valueOf(i);
           //System.out.println(indexID);
           indexRec.addLink(indexID);
           indexRec.addAttributes(tripSortedIndex[i],indexID);
        }

        //System.out.println("Sorted Trip Index: " + Arrays.toString(tripIndex));

        System.out.println("Write sorted trip index: sortedTripIndex.csv");
        try {
           CSVWriter writer = new CSVWriter(new FileWriter("sortedTripIndex.csv"), '\t');
           Iterator<linkRecord> itrCount = indexRec.iterator();

           while (itrCount.hasNext()){
              linkRecord odRec = itrCount.next();
```

```
            //System.out.println(odRec.getAttributes());
            String[] entries = odRec.writeToStrings();
            writer.writeNext(entries, false);
         }
         writer.close();

      }catch (Exception e) {
         throw new RuntimeException(e);
      }finally {}

      System.out.println("Write first 4000 ODs: sortedOD.csv");

      int maxOD = 3000;
      try (PrintStream out = new PrintStream(new
FileOutputStream("selectlink"+maxOD+".txt"))) {
         for (int j = 1; j <= maxOD; j++) {
            int odIndex = tripSortedIndex[tripRec.size() - j];
            //System.out.println(odIndex);
            String od = String.valueOf(odIndex);
            int originNum = tripRec.getAttributes(od).get(origin);
            int destNum = tripRec.getAttributes(String.valueOf(od)).get(destination);
            //System.out.println(originNum + " " + destNum);
            out.print("myA["+j+ "] = " + String.valueOf(originNum));
            out.println();
            out.print("myB["+j+ "] = " + String.valueOf(destNum));
            out.println();
         }
      out.close();
      }catch (Exception e) {
         throw new RuntimeException(e);
      }finally {}

      System.out.println("Write first 4000 ODs: selOD.csv");
      try {
         CSVWriter writer = new CSVWriter(new FileWriter("selOD.csv"), '\t');
         for (int k = 1; k <= maxOD; k++) {
            int odIndex = tripSortedIndex[tripRec.size() - k];
            //System.out.println(odIndex);
            String od = String.valueOf(odIndex);
            int originNum = tripRec.getAttributes(od).get(origin);
            int destNum = tripRec.getAttributes(od).get(destination);
            int odDemand = tripRec.getAttributes(od).get(trips);
            String odID[] =
{String.valueOf(originNum)+String.valueOf(destNum),",",String.valueOf(odDemand)};
            writer.writeNext(odID, false);
         }
```

```
            writer.close();
        }catch (Exception e) {
            throw new RuntimeException(e);
        }finally {}
    }
}
```

Java Script – tripMatrix.java

```
/**
 * Created by Yuchen Cui on 3/26/15.
 * This java class is used to identify irrelevant links
 */

import com.opencsv.CSVReader;
import com.opencsv.CSVWriter;
import com.pb.common.matrix.CSVMatrixWriter;
import com.pb.common.matrix.Matrix;

import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Random;

public class tripMatrix {
    public static void main(String[] args) {

        /*Random rand = new Random();
        for (int i = 0; i < 10; i++){
            if (rand.nextFloat()>0.8) System.out.println("Yeap");

        }
*/
        //define file names
        int maxOD = 2000;
        int v = 14;

        String linkFile = "linkIndex_"+v+".csv";
        String odFile = "selOD.csv";

        int A_B = 0;
        int O_D = 0;
        int aNode = 1;
        int bNode = 2;
```

```
        int asgnVol = 3;
        int volcap = 4;
        int index = 5;

        HashMap<String, Integer> linkMap = new HashMap<>();
        HashMap<String, Integer> linkVolMap = new HashMap<>();
        HashMap<String, Float> linkVCMap = new HashMap<>();
        int maxLinkIDNumber = Integer.MIN_VALUE;

        System.out.println("Read link file: linkIndex.csv");
        try {
           CSVReader linkReader = new CSVReader(new FileReader(linkFile));
           String[] nextLink;

           while ((nextLink = linkReader.readNext()) != null) {
              String linkID = nextLink[A_B];
              String linkIndex = nextLink[index];
              String linkVC = nextLink[volcap];
              String linkVol = nextLink[asgnVol];
              linkMap.put(linkID, Integer.valueOf(linkIndex));
              linkVolMap.put(linkID, Integer.valueOf(linkVol));
              linkVCMap.put(linkID, Float.valueOf(linkVC));
              maxLinkIDNumber = Math.max(maxLinkIDNumber,
Integer.valueOf(linkIndex));
        }
        }catch (Exception e) {
           throw new RuntimeException(e);
        }finally {}

        int[] linkArray = new int[linkMap.size()]; //store link index
        String[] linkABNames = new String[maxLinkIDNumber + 1]; //store link ID
        int position = 0;
        for (String linkId: linkMap.keySet()) {
           linkArray[position] = linkMap.get(linkId);
           linkABNames[linkMap.get(linkId)] = linkId;
           position++;
        }
        System.out.println(linkABNames[2]);

        System.out.println("reserve link index in an int[]");

        System.out.println("Read link file:selOD.csv");
        System.out.println("reserve OD index in an int[]");
        int[] odArray = new int [maxOD];
        int[] odDemand = new int[maxOD];
        try {
```

```
            CSVReader odReader = new CSVReader(new FileReader(odFile));
            String[] nextOD;
            int k = 0;
//change k range
            while ((nextOD = odReader.readNext()) != null && k < maxOD) {
               String odID = nextOD[O_D];
               //add OD as "i"+"j"
               odArray[k] = Integer.parseInt(odID.trim());
               odDemand[k] = Integer.parseInt(nextOD[1].trim());
               k++;
            }
         }catch (Exception e) {
            throw new RuntimeException(e);
         }finally {}

         System.out.println(odArray.length);

         System.out.println("Start to write to matrix for link volumes of selected OD");
         Matrix linkByOD = new Matrix(linkArray.length, odArray.length);
         Matrix tripShare = new Matrix(linkArray.length, odArray.length);
         linkByOD.fill(0);
         tripShare.fill(0);
         /*linkByOD.setValueAt(1,1,270);
         (linkByOD.getValueAt(1, 1)); */

         //set external numbers using linkArray[] and odArray[];
         linkByOD.setExternalNumbersZeroBased(linkArray, odArray);
         tripShare.setExternalNumbersZeroBased(linkArray, odArray);

         for (int k = 1; k <= maxOD; k++) {
            int odIndex = odArray[k-1];
            try {
               System.out.println("reading Links_OD"+k+".csv");
               String inputFile = "Links"+v+"_OD"+k+".csv";
               CSVReader linkReader = new CSVReader(new FileReader(inputFile));
               String[] nextLink;

               while ((nextLink = linkReader.readNext()) != null) {
                  String linkID = nextLink[A_B];
                  int linkIndex = linkMap.get(linkID);
                  int vol = Integer.parseInt(nextLink[3].trim());
                  linkByOD.setValueAt(linkIndex, odIndex, vol);
               }
            } catch (Exception e) {
               throw new RuntimeException(e);
            } finally {
```

```
            }
        }
        System.out.println(linkByOD.getSum());

        System.out.println("calculating tripShare.matrix");
        //List<Integer> linkList = new ArrayList<Integer>();
        for (int k = 0; k < maxOD; k++) {
            float totOD = odDemand[k];
            for (int j = 0; j < linkArray.length; j++) {
                float volLinkByOD = linkByOD.getValueAt(linkArray[j], odArray[k]);
                tripShare.setValueAt(linkArray[j], odArray[k], volLinkByOD / totOD);
            }
        }
        System.out.println(tripShare.getSum());

        CSVMatrixWriter linkByODMat = new CSVMatrixWriter(new
File("linkByODMat.csv"));
        linkByODMat.writeMatrix(linkByOD);

        CSVMatrixWriter tripShareMat = new CSVMatrixWriter(new
File("tripShareMat.csv"));
        tripShareMat.writeMatrix(tripShare);

//System.out.println(linkByOD.getValueAt(indexDatabase.getAttributeAt("4970_5066",
0), 359248));

        System.out.println("analyzing tripShare.matrix and write results to
resultMat.matrix");
        int[] catArray = {0,1,2,3};
        double cat0 = 0;
        double cat1 = 0.2; //initial trip share cutoff value
        double cat2 = 1;
        double maxVC = 0.5;
        double maxPct = 0.1; //1% of total links
        double inc = 0.0001; //increment of trip share cap
        System.out.println(String.valueOf(linkArray.length*maxPct));

        Matrix resultMat = new Matrix(linkArray.length, catArray.length);
        //set external numbers using linkArray[] and odArray[];
        resultMat.setExternalNumbersZeroBased(linkArray, catArray);

        while (cat1 < 1) {
            resultMat.fill(0);

            for (int k = 0; k < linkArray.length; k++){
                int linkID = linkArray[k];
```

```
            for (int j = 0; j < odArray.length; j++ ){

               int odID = odArray[j];
               double tripShareValue = tripShare.getValueAt(linkID,odID);
               //category 0: tripShare == 0
               if (tripShareValue == cat0){
                  float count = resultMat.getValueAt(linkID, catArray[0]);
                  count++;
                  resultMat.setValueAt(linkID, catArray[0], count);
               }
               //category 1:0 <=tripShare< cat1
               if (tripShareValue > cat0 && tripShareValue < cat1){
                  float count = resultMat.getValueAt(linkID, catArray[1]);
                  count++;
                  resultMat.setValueAt(linkID, catArray[1], count);
               }
               //category 2: cat1 <=tripShare< 1
               if (tripShareValue >= cat1 && tripShareValue < cat2){
                  float count = resultMat.getValueAt(linkID, catArray[2]);
                  count++;
                  resultMat.setValueAt(linkID,catArray[2], count);
               }
               //category 3: tripShare == 1
               if (tripShareValue == cat2){
                  float count = resultMat.getValueAt(linkID, catArray[3]);
                  count++;
                  resultMat.setValueAt(linkID, catArray[3], count);
               }
            }
            //System.out.println("count of 100% OD trip share: " +
resultMat.getValueAt(linkID,catArray[4]));
         }
         //System.out.println(resultMat.getSum());
         //CSVMatrixWriter resultLinkCat = new CSVMatrixWriter(new
File("linkByCat.csv"));
         //resultLinkCat.writeMatrix(resultMat);

         System.out.println("write out links to a list, control total number of irrelevant
links");
         List<String> excLinkList = new ArrayList<String>();
         for (int k = 0; k < linkArray.length; k++) {
            int linkIndex = linkArray[k];
            String linkAB = linkABNames[linkIndex];
            double linkVC = linkVCMap.get(linkAB);
            float countCat0 = resultMat.getValueAt(linkIndex, catArray[0]);
            float countCat1 = resultMat.getValueAt(linkIndex, catArray[1]);
```

```
            float countCat2 = resultMat.getValueAt(linkIndex, catArray[2]);
            float countCat3 = resultMat.getValueAt(linkIndex, catArray[3]);

            //find links that carries no more than cat1 TR for any OD
            if (countCat0 > 0 && countCat1 > 0 && countCat2 == 0 && countCat3 == 0
&& linkVC < maxVC) {
               excLinkList.add(linkABNames[linkIndex]);
            }
         }
         System.out.println("excLinkList size is " + String.valueOf(excLinkList.size()));

         if (excLinkList.size() < linkArray.length*maxPct) {
            cat1 = cat1 + inc; //update trip share cutoff point
            System.out.println("trip share cutoff point is " + cat1);
            try {
               System.out.println("write out excluded links to excLink.csv");
               CSVWriter writer = new CSVWriter(new FileWriter("excLink_"+v+".csv"),
'\t');
               for (int k = 0; k < excLinkList.size(); k++) {
                  String linkAB = excLinkList.get(k);
                  int linkVolume = linkVolMap.get(linkAB);
                  float linkVC = linkVCMap.get(linkAB);

               String excLink[] = {linkAB.split("_")[0], ",", linkAB.split("_")[1], ",",
String.valueOf(linkVC),
                        ",", String.valueOf(linkVolume), ",", linkAB};
                  writer.writeNext(excLink, false);
               }
               writer.close();
            }catch (Exception e) {
               throw new RuntimeException(e);
            }finally {}
         }
         else {
            System.exit(0);
         }
      }
   }
}
```

Cube Script – HwyAssign_Dissert_ODSplit.s

```
; Yuchen Cui's Dissertation
; Main directory: Models\NetworkDefiningModel\Assignment
; Trip assignment on *.net
; Version 1.0
;Yuchen Cui – 09/07/2014
```

```
maxIterns = 1000
DISTRIBUTE INTRASTEP= 1
prd = 'PM', spd='FFSPD', ConFac = 0.34, v = 10
READ FILE= parameter.dat

LOOP myCounter=1,2000
; OD coverage

RUN PGM=MATRIX
        zones=1
        ARRAY myA=2000, myB=2000
          ;change array length according to OD coverage

        READ FILE="selectlink2000.txt"

        toListA=myA[@myCounter@]
        toListB=myB[@myCounter@]
        LOG PREFIX=myMatrix VAR=toListA, toListB
ENDRUN

RUN PGM=HIGHWAY PRNFILE='Print\Highway Assignment_linkDis_oneInc_ODSplit.PRN' MSG='PM Highway Assignment for each OD selected'
        FILEI NETI = baltNetwork_run@v@.net

        FILEI MATI[1 ] = subareaVehPM_81zones.trp
          ;1 tables of 20 vehicle classes combined

          FILEO NETO = Output\DissertHwyAsgn_PM@v@_OD@myCounter@.tmp

        DistributeINTRASTEP ProcessID='HwyAssignIDP',ProcessList=1-20,
        MinGroupSize=4,  SavePrn=T
PARAMETERS ZONEMSG=20,MAXITERS=@maxIterns@, COMBINE=AVE, RELATIVEGAP = 0.02, GAP= 0, AAD=0, RAAD=0, RMSE=0

PROCESS PHASE=LINKREAD
        T0 = 60* (LI.DISTANCE/LI.FFSPD)
        C  = LI.CAPACITY*LI.LANE/@ConFac@ ; capacity
        LW.COSTa = T0 + (LI.TOLL_@prd@/@VoTa@) + 0.25*LI.DISTANCE
        ;five income groups combined

        ; Recode facility type (FUNCCLASS) into VDF groups.
        IF (LI.FUNCCLASS = 1,2,8,9,10)  LINKCLASS = 1     ; Freeway/Expwy & Ramps
        IF (LI.FUNCCLASS = 3,4)         LINKCLASS = 2     ; Arterial
```

```
        IF (LI.FUNCCLASS = 5,6,7)          LINKCLASS = 3     ; Collectors/Local
        IF (LI.FUNCCLASS = 11)             LINKCLASS = 4     ; Centroid Connectors

         ; Set link usage restrictions for this Highway Assignment.
         IF (LI.@prd@LIMIT = 4)  ADDTOGROUP = 1      ; no Trucks (MT or HT)
         IF (LI.DISABLED = 1) ADDTOGROUP = 2   ENDPROCESS

PROCESS  PHASE=ILOOP
;this PATHLOAD statement builds paths on TRAVEL COST, assigns each vehicle classV
        PATHLOAD PATH=LW.COSTa, EXCLUDEGROUP=2, VOL[1 ] = MI.1.1, MW[1] = MI.1.1,
         SELECTLINK = (A=@myMatrix.toListA@ && B=@myMatrix.toListB@), VOL[2 ]=MW[1 ]
ENDPROCESS

PROCESS PHASE=ADJUST
function {
   V=VOL[1]

   TC[1] = Min(T0 * (1 + 0.70*(V/C)^8), T0*100)
   TC[2] = Min(T0 * (1 + 0.55*(V/C)^6), T0*100)
   TC[3] = Min(T0 * (1 + 0.17*(V/C)^4), T0*100)
   TC[4] = T0
   }

  LW.COSTa=TIME + (LI.TOLL_@prd@/@VoTa@) + 0.25*LI.DISTANCE

ENDPROCESS
ENDRUN
ENDLOOP
```

Cube Script – HwyAssign_Dissert_Post.s

```
; Yuchen's Dissertation
; This script post process *.TMP to *.NET
; Main Directory: Models\NetworkDefiningModel
; Post processing after Highway Assignment

v = 10

LOOP myCounter = 1,2000

RUN PGM=NETWORK    PRNFILE='Print\PM Highway Assignment OD Split - Post Process.PRN'  MSG='PM Highway Assignment - Post Process'
FILEI LINKI[1] =  Output\DissertHwyAsgn_PM@v@_OD@myCounter@.tmp
```

```
        Vehicles     = V1_1
        SelTotal     = V2_1
        Total_Vol    = V_1
        VHT          = VHT_1
        VMT          = VDT_1
        AsgnCSPD     = CSPD_1
        VolCap       = VC_1
        CongTime     = TIME_1

 FILEO NETO = 81zones\Dissert_Veh_PM@v@_OD@myCounter@.net, EXCLUDE =
;Exclude indicated fields and combined income groups
          V1_1,V2_1,
                        V1T_1,V2T_1,
          VT_1, V_1, TIME_1, VC_1, CSPD_1, VDT_1, VHT_1, TOT_VOL
 ENDRUN
 ENDLOOP
```

GLOSSARY

Resolution – It is a geographic term applied throughout this dissertation. The definition given by the Data West Research Agency is: "It is a measure of the accuracy or detail of a graphic display. It represents the minimum difference or distance between two independently measured or computed values or objects that can be distinguished by the measurement or analytical method, or sensor being considered or used. In this study, it stands for how fine the geographic detail of an object is, such as the number of road links within the object or the size of the object." *(26)*

Geographic Information System (GIS) – A GIS is a computer system that can create, store, manage, edit, analyze, and display geographical data. The Esri's ArcGIS is a GIS tool that can create and edit maps, compile and manage geographical data in a geodatabase, and analyze geographical data in a range of tools.

Travel Demand Forecasting – It is a process of estimating the number of trips made using a specific transportation mode (e.g. automobile, truck, bus, and bicycle) on a specific route.

Traffic Analysis Zone (TAZ) – A TAZ is a geographic unit that is used to aggregate trips into a manageable area. The size of a TAZ varies depending on the analytical unit it needs to address. A TAZ could be as small as a single building; however the development of a finer TAZ system is often limited by data unavailability and increasing computational burden.

Transportation Network - A Transportation Network is composed of a series of links and nodes. A link represents a segment of street and contains attributes of the street

segment, and a node represents an intersection, or a point where network attributes are about to change.

Centroid and Centroid Connector – A Centroid is a special kind of node that is usually created at the center of gravity point of a TAZ. A Centroid connects its TAZ to the Transportation Network through a fictitious link called Centroid Connector. Usually a Centroid Connector is not allowed to connect to major arterial, freeways, or ramps.

Origin/Destination (OD) Trip Interchange – For a specific trip, its Origin is a zone where this trip begins; its Destination is a zone where this trip ends. A specific trip traveling from an Origin to a Destination with no intermediate stops represents a Trip Interchange.

Trip Table and Trip Matrix – For a given zone system, its Trip Table or Trip Matrix, which contains the OD Trip Interchanges, is a matrix of vehicles or persons traveling from one TAZ to another. Each row of the matrix represents trips traveling from one TAZ to all others while each column of the matrix represents trips coming to one TAZ from all others. Trip Interchanges in the Trip Table are loaded at Centroid and then assigned onto the Transportation Network through Centroid Connectors.

Travel Demand Model (TDM) – A TDM is a computer-based travel demand forecasting tool. A TDM usually follows the conventional sequential four-step process: Trip Generation, Trip Distribution, Mode Choice, and Trip Assignment. Citilabs' Cube is a computer software package developed for transportation modeling, containing several modeling modules and extensions.

Trip Purpose – It is common that people travel for different reasons. A TDM usually defines trips by Trip Purpose, such as home-based work, home-based school, home-based shopping, or non-home based.

Travel Mode – It can be generally classified into automobile, transit, walking and bicycling. Some TDMs define transit modes by access mode (automobile, walking, and bicycling) or by service type (local bus, light rail, and train) Some TDMs define automobile mode by occupancy level (drive alone, shared ride with two occupants, shared ride with three occupants, etc.).

Trip Generation – Its purpose is to estimate how many trips of each type that begin or end in each geographic unit. In a TDM, this step calculates how many vehicle trips or person trips by modes (e.g. automobile, walking, and bicycling) are produced and attracted in each TAZ. The output of Trip Generation is trip productions and trip attractions in each TAZ by Trip Purpose.

Trip Distribution – Its purpose is to estimate how many trips travel between zones. In a TDM, this step relates the trip productions and attractions from the Trip Generation step. The output of Trip Distribution is production-attraction Trip Tables by Trip Purpose.

Mode Choice – Its purpose is to split trips in the Trip Tables by Travel Mode. This step calculates how many trips between zones are made by each type of mode. The output of Mode Choice is Trip Tables by Travel Mode by Trip Purpose.

Trip Assignment – Its purpose is to routes vehicle trips and transit trips from the OD Trip Table onto a highway network and a transit network. This step calculates how many

trips take specific paths through a road or transit network. The result of Trip Assignment is traffic volumes on network links by time of day and by Travel Mode.

External Travel and External Zone – The External Travel refers to trips that begin and end in External Zones, which are TAZs outside a TDM's model area. In a TDM, its External Travel is estimated by Trip Generation and Trip Distribution and is usually represented in OD Trip Tables including External Zones and zones within the model area.

Intrazonal Trip and Interzonal Trip – An Intrazonal Trip refers to a trip whose Origin and Destination are within the same zone. An Interzonal Trip refers to a trip whose Origin and Destination are in two different zones. Only Interzonal Trips will be loaded onto the Transportation Network in the Trip Assignment.

Java – It is a computer programming language, which was originally developed by Sun Microsystems (merged into Oracle Corporation later on).

REFERENCES

1. Texas Transportation Institute. *The Effect of Network Detail on Traffic Assignment Results.* Texas Transportation Institute, Texas A&M University, 1967.

2. Chang, K., Z. Khatib, and Y. Ou. Effects of Zoning Structure and Network Detail on Traffic Demand Modeling. *Environment and Planning B*, Vol. 29, No. 1, 2002, pp. 37-52.

3. Giaimo, G. *Travel Demand Forecasting Manual 1 – Traffic Assignment Procedures* . Ohio Department of Transportation, 2001.

4. Bovy, P., and G. Jansen. Network Aggregation Effects upon Equilibrium Assignment Outcomes: An Empirical Investigation. *Transportation Science*, Vol. 17, No. 3, 1983, pp. 240-262.

5. Jansen, G., and P. Bovy. The Effect of Zone Size and Network Detail on all-Or-Nothing and Equilibrium Assignment Outcomes. *Traffic Engineering & Control*, Vol. 23, No. HS-033 448, 1982.

6. Khatib, Z., K. Chang, and Y. Ou. Impacts of Analysis Zone Structures on Modeled Statewide Traffic. *Journal of Transportation Engineering*, Vol. 127, No. 1, 2001, pp. 31-38.

7. JEON, J., S. KHO, D. KIM, and J. LEE. Interactions of Aggregated Zoning and Network Systems: A Case Study of Seoul City. *Journal of the Eastern Asia Society for Transportation Studies*, Vol. 8, 2010.

8. Jeon, J., S. Kho, J. J. Park, and D. Kim. Effects of Spatial Aggregation Level on an Urban Transportation Planning Model. *KSCE Journal of Civil Engineering*, Vol. 16, No. 5, 2012, pp. 835-844.

9. Haghani, A. E., and M. S. Daskin. Network Design Application of an Extraction Algorithm for Network Aggregation. *Transportation Research Record*, No. 944, 1983.

10. Ruddell, K., and A. Raith. Initializing the Traffic Assignment Problem by Zone Aggregation and Disaggregation. *Transportation Research Record: Journal of the Transportation Research Board*, No. 2466, 2014, pp. 52-57.

11. Connors, R., and D. Watling. Aggregation of Traffic Networks using Sensitivity Analysis. *UTSG, January*, 2008.

12. Chan, Y. A Method to Simplify Network Representation in Transportation Planning. *Transportation Research*, Vol. 10, No. 3, 1976, pp. 179-191.

13. University of Maryland College Park, and Parsons Brinckerhoff. *Maryland Statewide Transportation Model User's Guide.* Maryland State Highway Administration, Maryland, 2011.

14. Baltimore Metropolitan Council. *Travel Forecasting Model Calibration Report.* , 2006.

15. ———. *Baltimore Region Travel Demand Model for Base Year 2000 (Task Report 04-01).* , 2004.

16. National Research Council (U.S.). Transportation Research Board. *HCM 2010: Highway Capacity Manual.* Transportation Research Board, 2010.

17. Florian, M., and D. Hearn. Network Equilibrium Models and Algorithms. *Handbooks in Operations Research and Management Science*, Vol. 8, 1995, pp. 485-550.

18. Wardrop, J. G. ROAD PAPER. SOME THEORETICAL ASPECTS OF ROAD TRAFFIC RESEARCH. In *ICE Proceedings: Engineering Divisions*, Thomas Telford, 1952, pp. 325-362.

19. de Dios Ortuzar, J., and L. G. Willumsen. *Modelling Transport.* John Wiley & Sons, 2011.

20. Sheffi, Y. Urban Transportation Networks. , 1985.

21. Rose, G., M. S. Daskin, and F. S. Koppelman. An Examination of Convergence Error in Equilibrium Traffic Assignment Models. *Transportation Research Part B: Methodological*, Vol. 22, No. 4, 1988, pp. 261-274.

22. Boyce, D., B. Ralevic-Dekic, and H. Bar-Gera. Convergence of Traffic Assignments: How Much is enough? *Journal of Transportation Engineering*, Vol. 130, No. 1, 2004, pp. 49-55.

23. Baltimore Regional Transportation Board. *Maximize2040 Draft Plan.* Baltimore Metropolitan Council, Baltimore, Maryland, 2015.

24. Comsis Corporation., United States., Office of Highway Planning., Urban Planning Division., United States.,Federal Highway Administration.,. *Traffic Assignment, August 1973 : Methods, Applications, Products.* U.S. Dept. of Transportation, Federal Highway Administration ; For sale by the Supt. of Docs., U.S. G.P.O., [Washington, D.C.?]; Washington, D.C., 1973.

25. National Cooperative Highway Research Program. *Travel Demand Forecasting: Parameters and Techniques.* 716, Transportation Research Board, Washington, D.C., 2012.

26. Wood, S. J., and E. J. Wood. A Practitioner's Guide to GIS-Terminology. , 2000.