AN AGENT-BASED MODEL OF BICYCLISTS ACCESSING LIGHT-RAIL STATIONS IN SALT LAKE CITY

by

Joshua Andrew Groeneveld

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Geography

The University of Utah

August 2011

The University of Utah Graduate School

STATEMENT OF THESIS APPROVAL

The thesis of **Joshua Andrew Groeneveld**

has been approved by the following supervisory committee members:

Harvey Miller	, Chair	**6/3/2011** Date Approved
Thomas Cova	, Member	**6/9/2011** Date Approved
Reid Ewing	, Member	**6/14/2011** Date Approved

and by **George Hepner** , Chair of

the Department of **Geography**

and by Charles A. Wight, Dean of The Graduate School.

ABSTRACT

Many of the existing transportation analysis methods rely on aggregate data analysis. While these models have been well established in the transportation community, there are many components of transportation that these methods neglect. Among these marginalized components are multimodal trips. Using more than one mode of transport to get from origin to destination is not nearly as common as a single-mode trip, but multimodal trips can have a significant impact on the transportation system as a whole. Additionally, traditional analysis methods have difficulty capturing nuances in travel behavior that are best observed at an individual level such as trip-chaining.

This thesis shows how an agent-based model (ABM) can be used to analyze a multimodal transportation problem. Specifically, the model presented here describes the behaviors of bicyclists in Salt Lake City, Utah who use the TRAX light-rail system in conjunction with cycling to complete their trip. The model in this thesis includes the effects of elevation on mode choice, which is often ignored in traditional analysis methods. The model serves as a proof-of-concept that an ABM is a worthy means of analyzing multimodal trip patterns. The model is constructed in the NetLogo environment, which shows the usefulness of freely available software. Lastly, the outputs of the model show that an ABM can successfully capture trends that cannot be observed using traditional methods.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table Page

ACKNOWLEDGMENTS

I would like to thank my advisor, Harvey Miller, for his patience and helpful suggestions that made this project possible. His guidance has been extremely helpful, and he has helped me experience a realm of modeling I did not know existed before. Also, I am thankful for Tom Cova and Reid Ewing for serving on my committee and being patient with me as I experienced the ups and downs of creating a model.

I also owe an extreme debt of gratitude to David M. Johnson of the Flux Research Group in the University Of Utah School Of Computing. Without his persistent help and guidance understanding basic computer programming concepts, this project would not have been possible. If there was ever someone who deserved the title of "honorary geographer", David is it in my book.

Lastly, I would like to thank all of my friends and colleagues who continually encouraged me to see this project through to the end.

1. INTRODUCTION

In the 21st century, cities, municipalities and regional governments are attempting to plan communities which are no longer dependent upon the automobile. There is a great deal of focus placed on shifting trips from being auto-based to transit-based (Kuby *et al.*, 2004). Additionally, there is also a push to shift trips to nonmotorized or active modes of transportation, namely, walking and bicycling (Dill and Carr, 2003). Most of the existing methods for predicting transit ridership focus on unimodal trips, in which all of the total transportation trips either take place by automobile or by transit. One of the key components that some models treat superficially is the multimodal nature of transportation, especially when accessing public transit systems. Every person who rides transit must have a means of getting to the station or stop so that the transit service can be utilized.

Transit service is usually justified by the creation and subsequent performance of a travel demand model. Travel demand modeling is a very important practice as it relates to public transportation. As a form of public transportation, light-rail transit (LRT) demand models need to be produced to guide planning and transit agencies. Though traditional models are useful, they do not incorporate the multimodal nature of the entire origin-destination trip (TRB Special Report 288, 2007). Thus, this research seeks to generate a model for LRT trips which are preceded by bicycle trips.

There is a unique opportunity to examine travel demand via bicycle for LRT in Salt Lake City, Utah. The Spring 2009 issue of *Good* magazine listed Salt Lake City as one of the seven best bicycle cities in North America. *Good* magazine reported that 2.1% of the 180,000 people within the Salt Lake City limits commute by bicycle. Additionally, The League of American Bicyclists gives Salt Lake City a "silver" rating as a bicycle-friendly community based on the level of investment (from planning to engineering to education) that communities put into bicycle services (League of American Bicyclists, 2011). It is also possible for riders to bring their bicycles on TRAX at any time, as long as there is sufficient capacity at either end of any of the cars in the train. Currently, this is designated as two bicycles maximum at each location. Thus, modeling the use of bicycles for the access of TRAX patrons is useful to the Utah Transit Authority (who operates TRAX) and to other transit agencies who are hoping to entice more customers to use their bicycles to get to the transit station.

This thesis uses agent-based modeling (ABM) to simulate demand for light-rail among bicyclists in Salt Lake City, Utah. This research also uses a detailed public transit survey to evaluate the ABM. The results of this research will shed new light on the integration of bicycles and public transit usage. These results can also be used for "what-if" scenario planning by transportation planning agencies.

1.1 Background

Transportation models, usually those modeling the demand for a new or existing service, have been around for the past 60 years (Levy, 2000). A discrete choice model is

an existing model most similar to the model produced in this thesis. In discrete choice models, individuals choose transportation modes to complete a trip based on a set of discrete alternatives. Discrete choice models can be binary in nature (only two choice alternatives) or multinomial (three or more alternatives) (Ben-Akiva and Lerman, 1985). These models primarily consider trips taken via either automobile or public transit, though nonmotorized modes can be included (Ben-Akiva and Lerman, 1985).

An agent-based model is a model that consists of individual, autonomous entities (agents) that interact with other agents and the environment around them in ways that produce complex behavior. The basic premise of an ABM is that the individuals moving around in the model (the agents) have a set of fairly simple rules that govern their behavior as they interact with other agents and the space around them. As the agents execute these rules while moving and interacting, complex behavior arises (Flake, 1998). The complexity created and observed during an ABM simulation is not always a behavior that can be predicted beforehand. Thus, using an ABM allows modelers to observe behaviors which may be similar to those in the real world that would not be apparent based on observations from focus groups or travel diaries.

With respect to the model produced in this thesis, ABM is useful in more ways than just observing individual-level behavior. An ABM can also be coded to allow for multimodal trips. The ABM created here includes elevation as a variable directly influencing agent behaviors; this can be an important factor affecting nonmotorized travel in a hilly urban area such as Salt Lake City. Also, the ABM created for this project allows agents to follow rules posted on TRAX trains by UTA or to disregard these rules entirely. Although there is, theoretically, a flat cap on the number of bicyclists allowed

on any train at any given time, this rule is often disregarded in practice. An ABM can capture behavior that discrete choice models cannot, such as the bottleneck that occurs as cyclists attempt to board trains that are already at capacity (with cyclists). This particular behavior is unique because the outcomes are endogenous to the model, which means that the choice a cyclist makes (to ride TRAX or continue cycling) will only be made *after* the cyclist has already begun moving. Thus, it is not known a priori which specific choices each cyclist will make. ABMs are more suitable for these types of behavioral characteristics than discrete choice models.

ABMs and other advanced modeling techniques are resisted in the planning community due to a perception of high cost to implement (TRB Special Report 288, 2007). The model in this thesis utilizes the NetLogo modeling environment, which is freely available. The NetLogo platform reads GIS data, which will be very helpful to planning agencies since most of them have some GIS capability. This model serves primarily as a proof of concept; that is to say, it will show that using ABM is a worthwhile technique for this sort of a problem. It will also provide sufficient evidence that there are unique behaviors among cyclists in Salt Lake City that must be planned for so that UTA can maximize the potential ridership on TRAX among cyclists.

1.2 Research Objectives

The primary purpose for this research project is to develop an agent-based model (ABM) which shows behavioral characteristics of bicyclists in Salt Lake City, Utah accessing light-rail. This model is designed to capture behavioral characteristics inherent

in an individual's transportation decision-making that are unique to individuals using multiple modes of transportation over the course of a single trip. Additionally, this research intends to show that modeling nonmotorized travel (in this case, bicycling) is a tractable endeavor that can allow planning agencies to capture a more complete picture of the transportation choices individuals are making in their area.

Three objectives are outlined below that must be met for this project to achieve the stated goals above. These objectives were derived in general from the report produced by the Transportation Research Board Committee for the Determination of the State of the Practice in Metropolitan Area Travel Forecasting (TRB Special Report 288, 2007). The first objective is to show the usefulness of agent-based modeling. Since software is such an important consideration when modeling, the second objective is to use software at the disposal of existing planning agencies. The third and final objective is to show the usefulness of the simulation output results. While this is closely related to the first objective, it responds to a different barrier to advanced model implementation addressed by the TRB Committee.

As a means of testing the model to meet these objectives, three specific research questions are examined (see Section Four for the results of these experiments). The first research question is: how does the presence of elevation and major roads affect cyclist behavior? The second research question is: how does the variation of train headway and rule-following affect cyclist behavior? Finally, how does the variation of train headway and cyclist time waiting at the train station affect cyclist behavior? While these research questions are far from exhaustive, they do provide a good initial analysis of the effectiveness of the model.

1.2.1 Objective 1: Use of Agent-Based Modeling

The first objective of this thesis is to justify the use of agent-based modeling for capturing individual behavior in a multimodal context. Many planning organizations have not seen sufficient evidence that using a different type of model will provide necessarily better results or that the results of an advanced model can be explained if they are questioned (TRB Special Report 288, 2007). The Transportation Research Board convened a Committee for Determination of the State of Practice in Metropolitan Area Travel Forecasting to examine barriers to the implementation of newer models and delineate some of the weaknesses in existing modeling practices. Among the recommendations made by the committee regarding advanced modeling practices is a charge to develop newer models "that are better suited to providing reliable information for such applications as multimodal investment analyses…" (TRB Special Report 288, 2007).

One of the advantages of using an agent-based model is that an ABM can capture behaviors that are endogenous to the model itself, such as how cyclists proceed with their trip if a TRAX train is already at capacity with other cyclists. Since the cyclist's choice to ride the train while breaking the posted UTA rules is a dynamic decision (the cyclist is unaware of the bike capacity on the train ahead of time), this behavior cannot be accounted for or assumed ahead of time. Thus, a discrete choice model cannot adequately represent this decision making process. The purpose of this objective, then, is to show that an ABM can adequately represent both parameters that discrete choice models can examine (such as elevation, train headway, and the exclusion of some roads)

and parameters that discrete choice models cannot reasonably represent (such as rule-following behavior among cyclists).

1.2.2 Objective 2: Use of Available Software

Two of the TRB Committee's stated obstacles to advanced model implementation are cost and staff training for advanced models (TRB Special Report 288, 2007). Choosing a modeling package for an endeavor such as this involves many complex decisions, especially those related to the cost of software used to create the model and the cost in terms of labor to develop and run the model. Thus, the second objective of this project is to use software that most MPOs should already have (i.e., GIS) as well as software that is freely available for MPOs who desire to implement this type of model.

This model uses GIS data as part of the agent-based model. Using GIS data presents unique challenges in agent-based modeling, since the ability to read GIS data is not ubiquitous across all agent-based modeling platforms. Another design choice in this model is whether the agent-based software should be something tightly coupled with a GIS package, or should be a separate application which can also read and utilize GIS data. For the purposes of this model, the agent-based software is loosely coupled with GIS, which is to say that the application used runs separate from GIS software and can read GIS data entirely on its own.

1.2.3 Objective 3: Usefulness of Output Results

One of the other barriers the TRB Committee found that was preventing wide-spread adoption of advanced modeling practices is the "need for tangible evidence" that an advanced model is markedly better or improved upon existing models (TRB Special Report 288, 2007). Thus, the third objective of this project is to show that the variables measured in this model do affect individual behavior and that, if incorporated into a larger model, these variables would provide a more complete picture of overall travel demand in an area.

Planning agencies using travel demand models may not be terribly concerned with the new variables that an agent-based model can examine unless there is demonstrated evidence that the variables in an agent-based model produce significant changes in the results of model results. The model used in this project serves as a proof of concept, which means that it will not be very applicable "out of the box" to other study areas in other regions. While this model does seek to demonstrate the effectiveness of agent-based modeling, it will also show that variables which are captured in agent-based models (particularly those that cannot be included in discrete choice models) do have a significant impact on the output results. It will also show that these changes in output results would have major policy implications for planning agencies and transit agencies alike. The effects of the variables on simulation outputs are given in detail in the results section of this document.

1.3 Thesis Outline

This thesis has five major sections. The next section includes a literature review of the relevant aspects of travel demand modeling as it relates to the use of bicycles to access transit. This emphasizes areas such as the behavior of bicyclists in cities, LRT catchment areas, research incorporating a bike-rail connection, and traditional and newly-developed travel demand modeling methods. Following the literature review, the next section discusses the methods for creating a travel demand model for bicycle access to TRAX. This includes what specific methods will be used and why the selected methods are appropriate for this analysis. Within the methodology section, the data to be used for this study is briefly explained, both in terms of what is in the data and how it will be manipulated. The fourth section provides details on the results generated from simulated experiments. The analytical outputs to these results are then discussed. Finally, the fifth section provides discussion of what the results mean and how those results impact future research directions.

2. LITERATURE REVIEW

This section reviews five major areas that form the foundation of this thesis. The first subsection reviews factors influencing bicycle mode choice, since cyclists are of vital importance to the work in this thesis. This subsection includes a discussion on bicycle level of service, which shows how different roadways can be evaluated in terms of how well cyclists are served with bicycle-specific infrastructure. The second subsection examines factors influencing light-rail mode choice among transit users. The third subsection then looks at the combined bicycle and rail trip chain, since that is the focus of the multimodal analysis conducted here. Subsection four deals specifically with bicycles onboard trains as part of a multimodal trip. Subsection five is concerned with mode choice modeling, while subsection six deals with agent-based modeling.

2.1 Factors Influencing Bicycle Mode Choice

Since bicycle usage is generally small compared to automobile and transit modes, many of the policies directed toward bicycles are designed to increase bicycle ridership. In a very broad sense, bicycle mode share can be increased by creating policies and an environment which encourages bicycle use, while at the same time discouraging automobile use (Noland and Kunreuther, 1995). One such policy that encourages bicycle use is to make bicycling more convenient (Noland and Kunreuther, 1995; Dill and Carr, 2003). Many people do not use bicycles because they feel that it is not a viable option for

their commuting needs (Noland and Kunreuther, 1995). Other people do not use their bicycles because they feel that there is either insufficient bicycle infrastructure, or they feel that it would be unsafe (Dill and Carr, 2003; Moudon *et al.*, 2005).

One proposed means of increasing the level of bicycle usage has been to increase the amount of bicycle infrastructure available, such as dedicated lanes, paths, or even the use of bicycle detection at intersections (Dill and Carr, 2003; Moudon *et al.*, 2005). The results of increasing bicycle infrastructure are inconclusive. Dill and Carr (2003) suggest that infrastructure alone may not increase bicycle ridership, but new and improved bicycle facilities will be utilized and may be a contributing factor to more people using bicycles. Other work seems to point bicycle ridership as some form of self-selection, in that the factors causing people to choose bicycling are those which are not easily quantifiable (Moudon *et al.*, 2005). The presence of bicycle lanes and paths, along with motorized vehicle speed and presence of heavy vehicles, has been used in several studies that seek to understand the level of bicycle ridership (Dill and Carr, 2003; Landis *et al.*, 1997; Dixon, 1996). However, some subsequent studies have concluded that the presence of bicycle lanes is either insignificant, or the models used in previous studies were not statistically valid (Moudon *et al.*, 2005; Moudon and Lee, 2003). Thus, it would appear that though the physical environment may enhance a rider's cycling experience, it may not be the determining factor causing him or her to get on the bike in the first place.

In the past 15 years, some research has been done in terms of developing what is known as a bicycle level-of-service, or BLOS. This concept has been derived from the level-of-service (LOS) which was used for automobile transportation in the *Highway Capacity Manual* (Landis *et al.*, 1997). A level-of-service is designed to analyze the

quality of service a segment of road or other transportation link provides to its users. When specifically applied to bicycle usage, an LOS model measures user perceptions of hazards, rather than other performance measures, such as average speed or average time of delay (Landis *et al.*, 1997). Bicycle level-of-service can provide a distinct picture of how well a particular road serves cyclists. However, BLOS does not explain the factors driving individuals to choose to commute on a bicycle in the first place. Although BLOS may have some influence on people choosing to ride bicycles, it does not fully account for all aspects of a bicycle trip. Several studies on bicycle users note that not all cyclists have the same abilities, and therefore, not all cyclists feel as comfortable on the same section of road (Landis *et al.*, 1997; Dixon, 1996; Noland and Kunreuther, 1995). BLOS accounts for these differences in cycling abilities; the grade of BLOS a road segment receives provides a relative indication of how experienced potential bicycle users would be (Dixon, 1996; Landis *et al.*, 1997).

2.2 Factors Influencing LRT Mode Choice

There are several possible modes individuals can use to access light-rail stations, such as driving, feeder bus, walk and bicycle. Regardless of which mode an individual uses to get to the LRT station, there is *some* factor, or combination of several factors, which cause people to choose LRT to get from their origin to their destination. It is certainly possible that some of the weight behind the decision to use LRT is based on improvements in overall LRT performance and adherence to the needs of customers. Some of these improvements, many of which have gained popularity in the past decade,

include the adoption of low-floor vehicles by many transit agencies, raised station platforms and the ability of light-rail trains to "trip" traffic lights so that the train receives priority signaling at intersections (Topp, 1999). In spite of many of these improvements aimed at enhancing customer comfort, it is likely that the decision for most people to board light-rail trains stems from a variety of other factors.

In traditional analysis of LRT boardings, one of the major variables has been whether or not a particular station lies within the central business district (CBD) of a city. Stations within the CBD generally generate much higher daily boardings than their non-CBD counterparts. However, Kuby *et al.* (2004) found that including a dummy variable for presence in the CBD did not have a significant impact on the model used in their analysis. Kuby *et al.* (2004) hypothesize that the CBD variable is insignificant because the parameters which it is expected to capture in the model (such as high density land-use, high density employment, and so forth) are captured quite effectively using other variables. Thus, though there are potentially some factors in CBD's that are not included in most LRT travel demand models, the large influence of the CBD on daily boardings can generally be accounted for using other variables (Kuby *et al.*, 2004).

2.3 The Bicycle and Rail Trip Chain

When people utilize LRT, it is almost always as part of a trip chain (Givoni and Reitveld, 2006). The combined use of bicycles and rail transportation to form a trip chain is known as bike-and-ride (Martens, 2007; Martens 2004; Bracher, 2000). Bike-and-ride exists in several different forms. The most common form is to allow rail passengers to

ride a bicycle to and from rail stations by providing some sort of bicycle parking at the rail stations, usually in the form of bicycle lockers (Martens, 2007). Another form of bike-and-ride allows passengers to take their bicycles with them on the train, though this is less common in areas of Europe and Asia where bike-and-ride is popular (Martens, 2007; Bracher, 2000). Planning agencies have begun to realize the importance of bike-and-ride, and some city and regional governments are creating policies that are conducive to increasing the share of passengers who use bike-and-ride (Martens, 2007; Rietveld, 2000).

In a historical sense, the use of bike-and-ride in the United States is much lower than it is in some European countries (Replogle, 1992; Martens, 2007). Much of this can be attributed to the extensive emphasis on park-and-ride at American rail stations (Replogle, 1992). This emphasis on park-and-ride is attributed to the higher propensity of Americans to drive from place to place, as well as the nature of American cities which are sprawling and less dense than their European counterparts. However, as American city planners are seeking to design more sustainable and healthy cities, bike-and-ride is becoming more popular.

In Europe, one of the most popular places for commuters to use bike-and-ride is the Netherlands (Martens, 2007; Martens, 2004; Keijer and Rietveld, 2000). Bicycle usage overall is quite high compared to the rest of the industrialized world (Martens, 2007), which places the Dutch in a unique position to create policies to shift more trips from other modes to bicycles. Keijer and Rietveld (2000) found that just over 1/3 of all multimodal trips where rail is the primary mode use bicycling as a secondary mode. However, few rail lines in the Netherlands allow the bicycle to be brought onto the train,

causing the share of bicycle trips leading away from the activity-end train station to be much smaller (Keijer and Rietveld, 2000; Rietveld, 2000; Martens, 2007). The promotion of bike-and-ride can be attributed to many successful planning policies, but the most successful of these policies is the provision of secure and sufficient bicycle parking at rail stations (Martens, 2007). Since safe routes by bicycle to rail stations were available before much of the bicycle parking became available, it seems that merely increasing bicycle parking will promote bike-and-ride and can even lead to an increase in overall rail ridership (Martens, 2007). This agrees with the bicycle analysis by Dill and Carr (2003) in the United States which suggests that building new bicycle facilities (or improving existing ones) will cause more cyclists to use them.

Distance from the home-end rail station is also a factor in bike-and-ride. Martens (2004) notes that bicyclists will travel longer distances to reach faster modes of transport. The distance decay factor is a strong in the case of bicycle access to rail stations, in that most cyclists travel less than 3.5 km to reach the home-end station (Keijer and Rietveld, 2000). In addition to distance, density is also a factor in terms of bike-and-ride, since there needs to be sufficient residential density surrounding a rail station to make bike-and-ride viable (Martens, 2007; Martens, 2004).

2.4 Bicycles on Trains

A less common form of bike-and-ride allows passengers to bring their bicycles with them on the train (TCRP Report, 2005; Bracher, 2000). Though less common, this form of bike-and-ride seems to be gaining popularity, as transit agencies are beginning to

view it as a means of attracting more customers (Bracher, 2000). Those transit agencies which allow bike-on-rail services generally allow passengers to hang bicycles from hooks at designated bicycle areas within the rail car, or the bicyclist must stand with his bicycle in a designated area where there is sufficient room (TCRP Report, 2005; Bracher, 2000). The Utah Transit Authority is one such agency that allows for bicycles on their trains, including both the TRAX LRT vehicles as well as the FrontRunner commuter rail trains.

One of the conditions many transit agencies require of those bringing bicycles on trains is that those bicycles only be brought on the train during specified times, usually nonpeak travel hours (TCRP Report, 2005; Bracher, 2000). In a 2004 survey of major transit agencies which operate light rail, 10 out of 21 (48%) reported no time restrictions for when bicycles are allowed on trains (TCRP Report, 2005). Most transit agencies that enforce a time restriction do so to ease congestion within the rail cars, allowing more passengers to take up space which could otherwise be occupied by a bicycle (TCRP Synthesis 62, 2005). UTA enforces no time restriction on its TRAX trains, which is quite useful to the large number of University of Utah students who opt to bring their bicycles with them to campus.

2.5 Mode Choice Modeling

Discrete choice modeling has been applied to the mode-choice element of travel demand for many years. A discrete choice model analyzes the probability of an individual or group of individuals selecting an alternative from a finite group of possible alternatives (Ben-Akiva and Lerman, 1985). All forms of discrete choice models (such

as logit, probit and nested logit) assume that the individual or group of individuals being modeled all seek to maximize their own utility, which is to say that people will choose the mode of transport which gets them to their destination the quickest and for the least cost (Ben-Akiva and Lerman, 1985). Discrete choice models exist in both binary (only two choices) and multinomial (multiple choices) forms. Although the basis for discrete choice modeling occurs at the individual level, most of these models are aggregated to reflect groups of individuals (populations), since group-level decisions are more closely tied to policy considerations (Ben-Akiva and Lerman, 1985).

Discrete choice modeling as applied to mode choice typically models two alternatives: private automobile and public transportation (Ben-Akiva and Lerman, 1985). While the choices available to individuals in discrete choice models seem fairly simple, the factors underlying those choices are not simple. The two most important criteria for an individual's mode choice are quality of service (time) and cost (Levy, 2000). Discrete choice models use demographic information (including household income and race) about individuals in a study region to better predict which mode individuals will choose (Levy, 2000). Discrete choice models treat nonmotorized modes superficially if they are included at all. Since nonmotorized trips comprise such a relatively small share of modal split in the U.S., many discrete choice models choose to ignore them altogether (Ben-Akiva and Lerman, 1985).

Generally speaking, discrete choice models have been applied in situations where the subjects of analysis (individuals or groups) only make a choice once, at the beginning of the modeled time horizon. In many cases, this is a fair assumption because scores of people only use one mode of transportation to complete a trip. In other cases, though,

this assumption is not accurate when people make a multimodal trip. If a multimodal trip is one of the alternatives in the set of choices in a discrete model, then some mutli-modal behavior may be observed. A dynamic discrete choice model, where subjects make decisions more than once during a time horizon, can also capture such behavior (Brownstone, 2001). When discrete choice models do consider multimodal trips, time-based attributes have more significant impacts on choices than comfort-based attributes (Molin and van Gelder, 2008).

2.6 Agent-based Modeling

Agent-based models follow the object-based model paradigm. As such, there are certain elements that are characteristic of all objects (agents) within the model. In any object-based model, each individual object must be identifiable, relevant to the model, and describable using certain properties or characteristics (Worboys and Duckham, 2004). Additionally, an agent must have some sort of an end goal or final state that it is trying to attain (Miller, 2007). Agents are also capable of interacting with other agents, potentially causing the state of another agent to change, or causing the other agent to change its own state (Brown *et al.*, 2005; Miller, 2007). The agents are self-directed (Brown *et al.*, 2005), allowing them to pursue their own unique end goals separate of the other agents in the model. Also, the interactions between multiple agents are governed by a set of relatively simple rules (Brown *et al.*, 2005; Miller, 2007). Agents have some form of learning ability, allowing them to adapt to their environment over time (Miller, 2007; Kikuchi *et al.*, 2002). This learning ability allows them to learn that cooperating

with other agents will allow them to solve complex problems and create complex structures (Kikuchi *et al.*, 2002; Miller, 2007). Learning allows agents to constantly change how they respond to various inputs, which is analogous to how humans respond to various changes in their environment (Kikuchi *et al.*, 2002).

As agents continue to move and interact with the environment around them, the complexity they create can lead to a sort of global order, known as self-organization (Flake, 1998). This has interesting implications for applying ABM to transportation problems. It is possible that as people move around in space, their movements become increasingly dependent on the movement of others. As an example for this research project, a bicyclist who wants to ride TRAX at a specific time each day may be persuaded to find alternate means to getting around if she finds that the train is particularly crowded with other cyclists at that hour. In an ABM, the complex order is observable, and the rules are known, yet it is not necessarily possible to predict the future state of the system (Flake, 1998). The inability to see unexpected future outcomes based on known rules is one of the primary reasons for using ABM in this research: even though the rules are known, the outcomes are not.

Since travel occurs in a real-world space, it is crucial to integrate GIS data and processing with ABM. Many ABMs do not occur in a real-world space. These models examine complexity in some sort of proprietary space. Many of the available ABM packages did not support GIS data until the last 5 or so years. Those packages that had integrated GIS data into their models had trouble initially to take advantage of the power of a GIS (Gimblett, 2002). For the ABM packages available currently, the vast majority of them are stand-alone packages that can read GIS data but have limited interaction with

the GIS itself. Integrating GIS data into an ABM allows much more realistic spaces for agents to move around in, which allows their movements to be more similar to those of humans (Gimblett, 2002). As agent movement is more human-like, it is then more useful for decision-makers using ABM for planning purposes (Gimblett, 2002).

2.7 Conclusion

This section examines major topics germane to this research project, namely factors influencing people to ride a bicycle, factors influencing people to use light-rail and the connection between the two. Additionally, this section reviews existing work done in the field of travel demand, both as it applies to bicycles and to rail travel. This includes advanced modeling techniques in the discussion of activity-based analysis and agent-based modeling. The next section delineates the specific choices made in the creation of the ABM, from the path-finding algorithm used to the rules imposed on each cyclist.

3. METHODOLOGY

This section discusses the components involved in building the ABM. Although the NetLogo environment is available "off the shelf", the model used for this project required the use of supplementary GIS data to provide the model with a more realistic spatial representation. The GIS data were obtained from the Utah Automated Geographic Reference Center (AGRC) via their website. Additionally, the data used required some modifications before it could be imported into NetLogo. The first subsection of this chapter discusses the study area of the model and the rationale underlying the area chosen. This is followed by a detailed discussion of the data underlying the model and the modifications made to the data. The third subsection discusses how the model was constructed in terms of translating GIS data based upon the real world into a NetLogo space. The fourth subsection describes the path-finding algorithm for this model and how the algorithm was specifically adapted for use in NetLogo. The fifth subsection describes the parameters used in the model, and allowable values for each variable. The sixth subsection delineates the rules governing the movement of cyclists as they proceed from origin to destination. These rules are then presented as a decision tree the cyclist follows as it traverses its path, showing that decisions are made dynamically rather than statically at the beginning of each simulation.

3.1 Study Area

The study area of this research is Salt Lake City, Utah. Currently, all of the light-rail lines operated by the Utah Transit Authority (UTA) are within Salt Lake County. The extent of the light rail network covers most of the county in the north-south direction, and at present, only a small portion of the county in the east-west direction. In 2006, a comprehensive transit survey was conducted by NuStats on behalf of the Wasatch Front Regional Council. The model for this thesis is tested against this dataset (see Section 4.4). The NuStats dataset (see Section 3.2) covers all of Salt Lake County, and includes portions of Utah County to the south, as well as Davis and Weber Counties to the north. Since there is bus service outside of Salt Lake County all responses in the survey with an origin outside of the Salt Lake County will be removed from the analysis. In April 2008, UTA began a commuter rail service running between Salt Lake City and Ogden, Utah, which is in Weber County. The commuter rail service also allows bicycles on board, but it was not constructed at the time of the survey.

The focus of this model is the University TRAX line, which runs from downtown Salt Lake City east toward the University of Utah. There are four TRAX stations on the University campus: one at Rice-Eccles Stadium, one near the Huntsman Center (basketball arena), one near Fort Douglas, and the final station on the line is at the University's Medical Center. At the time of the survey, the western terminus of the University line was located in downtown Salt Lake City at Energy Solutions Arena. The University line also stops at the Salt Lake City public library and near the Trolley Square shopping district.

The justification for focusing solely on the University line for this model is based on making the simulations run more quickly. Since this model is more in line with a proof-of-concept rather than a citywide model that could be used by transit planners off the shelf, focusing on the University line (which is the shorter of the two TRAX lines, with the other being the Salt Lake-Sandy line) makes the model more tractable. The main hill going east from downtown up to the University only affects the University line, so terrain can still be included in the model. Additionally, it is much easier to visualize the University Line in NetLogo than it is to visualize the entirety (or vast majority) of the TRAX network and the Salt Lake County road network. So, constructing a model over a smaller area allows the model to be more tractable for simulation runs.

Figure 3.1 shows the study area used in the ABM. The TRAX stations shown are those stations that were on the University Line in 2006, which reflects the WFRC data (see subsection 3.2). Since Keijer and Reitveld (2000) found that cyclists will not ride more than roughly 2 miles (3.5 km) to reach rail-based services, a simple buffer was applied in ArcGIS to the University Line that selected all roads within 2 miles of any point on the line. The selected roads were then classified as major or nonmajor roads. Major roads were all of the designated state highways or U.S. highways. (U.S. 89 is State Street, a busy thoroughfare connecting the Utah State Capitol with the rest of the county going as far south as Draper.) Interstates were excluded from the roads in the study area since bicycles are not allowed to ride on them. See subsection 3.3 for a complete discussion on how the ABM was developed based on the study area GIS data.

3.2 Data Used in the Model

The data used in the model come from a transit survey conducted by NuStats in 2006 on behalf of the Wasatch Front Regional Council (WFRC). The NuStats survey was given to transit riders on the buses and light-rail trains run by the Utah Transit Authority. The survey asked participants trip related questions, such as the origin and destination of the trip, and how they arrived to their first transit vehicle and how they arrived at their destination from their final transit vehicle. The survey also noted each route taken for the participant's particular trip. The survey also asked various demographic questions of the participants. Of the more than 4,000 responses to the survey, 120 participants used a bicycle to get to their first transit trip. Though this number seems fairly small, representing roughly 2.5% of the survey respondents, it does come close to the percentage of bicycle commuters in Salt Lake City as reported in the Spring 2009 issue of *Good* magazine (2.1%).

The GIS data used in this model are all based off of data available from the Utah AGRC. Three primary datasets were obtained directly from the AGRC website: a point shapefile of light rail stations, a line shapefile of light rail lines, and a line shapefile of Salt Lake County street centerlines. While provided by AGRC, the light rail data were produced by UTA and the street centerlines were produced by the Salt Lake County Surveyor's Office. It was necessary to modify all of the GIS shapefiles so that the proper representation could be created in NetLogo.

The TRAX stations represented in the model had to be reduced to only those stations along the University Line which were in use in 2006 (to mimic the WFRC data discussed above). The GIS data included all stations across the TRAX network, so those

stations belonging solely to the Salt Lake City – Sandy Line were removed. Several additional nodes were added to the TRAX station network as well. In NetLogo, using point-based GIS data to create a set of agents places agents only at those points represented in the dataset. Creating links (or network arcs, see discussion in Section 3.3) from GIS data is a bit more challenging. Links are only created between two nodes that represent end-points of line segments in the GIS data. Thus, if the only TRAX stations used in the station network are those points where trains actually stop, the resulting NetLogo set of TRAX lines will look unrealistic, with links passing through buildings and other off-network places. In order to get the NetLogo network to mimic the real-world, a few additional pseudo-stations were added along curved points of the network. All of the TRAX lines and streets in the NetLogo network are straight lines, so a curved segment is portrayed as a collection of short, straight line segments. Attributes were also added to the GIS data, since NetLogo can read shapefile attribute tables. This allowed all of the stations (and nonstation nodes) to have both eastbound and westbound numbers so that the trains traveling over the TRAX lines would stop at each station in successive order.

For the linear TRAX line data, a few adjustments needed to be made in terms of where a line segment started and ended. For NetLogo to properly re-create the data, all of the segments had to be split up such that segments started and ended at the location of an existing TRAX station or a nonstation node. Special care was taken for the portion of the TRAX line covering the intersection of Main Street and 400 South, where the University Line diverges from the Salt Lake City – Sandy Line. In the original GIS data, the line segment that was supposed to connect two portions of the turn off of Main Street

and onto 400 South stopped short, leaving two separate, unconnected segments. These segments were snapped together using the snapping feature of the Editor toolbar in ArcGIS. The original dataset also included portions of the TRAX line west of the Arena station headed toward the Salt Lake City Intermodal Hub. Since that linear extension opened in mid-2008, those line segments were removed from the dataset.

The street centerline data required the most work to make the data compatible with NetLogo. Since NetLogo would only re-create streets between endpoints of a line segment, it was necessary to create a point GIS dataset of street intersections. A point dataset was developed by importing the street centerlines into an ArcGIS Network Dataset inside of a file geodatabase. One of the by-products of creating a network dataset is the creation of a point feature class of network junctions, which represent street network intersections. This point feature class was exported to a shapefile, since NetLogo does not yet support reading data from geodatabases. The portions of Interstate 15 and Interstate 80 falling within the study area were removed from the road dataset since cyclists were not allowed to travel on the interstate.

Several additions were made to the street network and street intersections. Since one of the primary goals of the model was for cyclists to access TRAX stations, additional roads and intersections had to be created such that a street intersection was located at the same point (topologically coincident with) a TRAX station. In reality, all TRAX stations are accessible by road, but this is not reflected in the GIS data, that show the TRAX network and the street network as two mutually exclusive entities. While many TRAX stations are realistically accessible from both ends of the platform (from two streets), it was not possible to model this in the NetLogo environment. The cyclists'

path-finding algorithm (discussed in section 3.4) created a loop when a cyclist arrived at a station accessible from both ends, causing the software to return an error. Thus, all TRAX stations are accessible in NetLogo from only one end of the platform, representing the street intersection closest to the station. Attributes were added to the street intersection datasaet, representing the name of the closest TRAX station to any given intersection as well as whether or not the intersection was located at a TRAX station. Thus, while all of the GIS data used to create this model are freely available, the model could not have been created in the bounds of the study area without some slight modifications of the GIS data.

The final piece of data used in this model is the raster-based elevation data. Elevation is one of the key components of the model, and NetLogo's ability to read raster ASCII files allowed the representation of hills in the model. The elevation data, also provided by the AGRC, are derived from 1.25 meter Light Detection and Ranging (LiDAR) data collected during 2006. The LiDAR data were available in raster tiles on the AGRC website, which were then stitched together in ArcGIS. Within NetLogo, the elevation for each grid cell was accessible to the street intersection nodes, which was used to calculate the slope for each street.

3.3 Construction of the Model

The model is developed in the NetLogo software package. NetLogo is a freely available agent-based modeling toolkit. There are several reasons why using NetLogo makes sense for this application. First, aside from low cost, NetLogo models have an

intuitive programming language. Although the engine running NetLogo operates based on Java, the code the user creates to implement the model is proprietary to the software. This allows users with a varying range of programming experience to create equally valid and useful models. Since programming experience is not necessary to create a NetLogo model, this makes it more accessible to those interested in what the model can do rather than the model itself.

Since the release of version 4.0.4, NetLogo supports GIS data. This functionality is crucial to this model. This model is simulating the movement of bicyclists on actual Salt Lake City roads and TRAX lines. With this in mind, GIS shapefiles available freely from the Utah Automated Geographic Reference Center (AGRC) are used to build the network the agents traverse in the model. In this model, the agents are not actually traveling over the GIS data itself. Rather, the GIS data are used as the basis for creating objects in the NetLogo world. Additionally, this extension makes the model more reproducible, since the GIS data used, including the road network, TRAX stops and TRAX lines, are all available from the Utah AGRC website.

In NetLogo, there are three classes of agents: turtles, links and patches. In this model, there are four types of turtles: nodes, stations, people (bicyclists) and TRAX trains. Turtles are the agents which actually move around within the NetLogo model. People represent bicyclists who have an origin and destination and find a path between the two points. Trains represent TRAX trains moving along the University Line, and they may carry people from one station to another. Although turtles can move, they do not always move. The nodes in this model represent the intersections of streets within the Salt Lake City study area. Stations represent the location of TRAX stations along the

University Line. Since intersections and stations are stationary, these turtles do not move in the model. The nodes also serve as origins, destinations, and intermediate points for the people. If the person decides to take TRAX, the stations also serve as intermediate points between the person's origin and destination.

Links are agents that connect one or more turtles together. In this model, the links that are visible are those links that connect a node with other nodes or a station with other stations. The people move from one node to another with their movement being constrained to the link neighbors (other nodes) of the current node they are located on. Similarly, trains can only move from one station to the next based on which station is linked to the station where the train is currently located. The links between nodes (streets) are directed. Two-way streets are created in the model by created a directed (one-way) link between two nodes in both directions. One-way streets are created by establishing a directed link only in the direction of travel. The one-way streets in the model are connected directly to the one-streets in reality. The GIS data from the Utah AGRC have a one-way attribute, and this attribute was used in the creation of the streets during the model setup process. Although there may be a few one-way streets in reality that are represented by two-way streets in the model, the GIS data correctly has the portions of 500 South Street and 600 South Street marked as one-way. These two one-way streets are the most important one-way streets in this model because they are both heavily used for automobile traffic. The links representing TRAX lines are also directed, which means that a TRAX train traveling from one station to another must visit them in order. This is most sensible for train movement because a train will visit each stop along the line in the same order for every trip. Each station is assigned a number in both the

eastbound and westbound directions. Lists are created based on these directional orders, and the moving trains then access these lists and move to the next station on the list. For the TRAX lines, two directed links are created between each station (one in each direction).

Patches represent the space over which the turtle agents move. The world is divided into square cells of equal size, and each cell represents one patch. Patches can never move in a model, but they can possess attributes, such as color. In the NetLogo world, patches are the "ground," which implies that all of the turtles and links exist on top of the patches. NetLogo allows flexibility from a topological standpoint such that patches on the edge of world can be connected to patches on the other edge. (In this case, the world would resemble a torus, where each patch would have exactly eight other patches it shares either a side or a corner.) The user can also constrain the topological properties of the world to allow "edge-wrapping" on left and right sides, the top and bottom sides, both or none. For the purposes of this model, edge-wrapping is not activated so that no cyclist moves across the world in a way physically impossible in reality.

The GIS data integrate into the model through the patches. Although the GIS data can be displayed in the model, it cannot be used directly to constrain the movement of the turtles. The core function of the GIS extension used in this model is the intersect command. The intersect command works in NetLogo the same way that it does in a standard GIS software package. The agents are created by asking the GIS extension to find the patches which intersect the GIS data, and then the patches spawn, or "hatch" the desired agent. When the patches hatch an agent, it is hatched at the center of that patch.

Thus, all of the nodes and stations appear at the center of each patch they intersect, even if the GIS data is not in the center of that patch. NetLogo does allow the patch size to be altered, but the patches still must be of some reasonable size so that everything running in the model is visible. The streets and TRAX lines are then created to link adjacent nodes or stations based on the GIS data. (This allows the streets and TRAX lines in the model to mimic the streets and TRAX lines in reality without creating any unnecessary or nonexistent ones.)

The primary reason that people traverse the network by moving between nodes that are connected by streets is that this implementation allows the path-finding of the people to be reduced to solving a graph. It is possible in NetLogo to cause agents to move using other criteria, such as setting patches intersecting roads to a particular color and then constraining the movement of the agents to be across only those patches of the color of the roads. If the road network was strictly a grid where all roads had a heading of 0, 90, 180 or 270 degrees, this would work very well; however, since the Salt Lake City road network is not entirely a grid, using a graph to find paths for the people is simpler and allows the model to run much quicker. Once the route-finding is reduced to a graph, one of many algorithms can be used to solve the graph based on any number of criteria.

This model makes several assumptions for the criteria used to find routes from origin to destination. First, this model presumes that all bicyclists will travel along the shortest, or least-cost, path. Second, this model assumes that speed is discrete, rather than continuous. The elevation variable determines directly how quickly a person can traverse one street segment. Elevation at each end of a street determines the slope of that

street, and the slope then determines cyclist speed. Since two-way streets are represented by two directed links (one in each direction), the speed of the uphill link will be different than that of the downhill link. The slope of each street is grouped into several speed categories. Cyclist speeds based on street slopes are shown in Table 3.1.

For TRAX lines, elevation is not used to determine the cost of traversing a link. Several factors influence the speed of TRAX trains. In reality, TRAX trains reach a maximum speed of 55 miles per hour on the Salt Lake – Sandy Line, but this is a result of few at-grade intersections with roads and longer distances between stations. On the University Line, TRAX trains do not have signal priority as they do on the Salt Lake – Sandy Line. Thus, TRAX trains may have to wait at an intersection for automobile traffic to clear before it has a clear signal. The maximum speed on the University Line is 35 miles per hour, but it is rare for trains to maintain this speed for very long given the proximity of the stations and the time spent waiting at intersections.

In this model, the time TRAX trains spend waiting at intersections is ignored, but this wait time is factored in at other places. Based on a University Line timetable obtained from UTA, it takes between 23 minutes to complete the trip from the Arena station to the University Medical Center station. The reverse trip takes 24 minutes, assuming that the train is on schedule. This timetable was current as of 2010, but this was deemed appropriate for this model since no major changes in infrastructure have occurred between 2006 (the time period represented in the model) and 2010 when coding took place. The model assumes a worst-case on-time scenario of each TRAX trip taking 24 minutes from end to end. The length of the University Line from the Arena station to the University Medical Center station is a total of 25,714 feet, based on the GIS data

obtained from the AGRC. This distance is roughly 5 miles (4.87 mi). The wait time for each TRAX train is factored in at each intermediate station on the train's trip. In reality, dwell time, or the time spent a given station, can range from 30 seconds to 2 or 3 minutes, based on personal observation. Dwell time in the model is 60 seconds (1 minute) at each station, for the sake of simplicity. Including both ends of the University Line, there are 11 TRAX stations. For a complete trip from end to end, there are 10 intermediate stops on this one-way trip, if the first end station is included (as it is in the timetable) as a timed stop. Thus, each train spends 14 minutes of each one-way trip actually in motion (24 minutes of total time – (10 stations * 1 minute of dwell time per station)). The speed at which each train moves is then: 25,714 feet (total distance) / 14 minutes (time spent moving, speed > 0) = 1,836.7 feet per minute, or 20.86 miles per hour. Since acceleration and deceleration of the moving trains is not factored in the model, 20.86 miles per hour seems like a reasonable speed for most trains most of the time.

3.4 Path-finding Algorithm

It is possible to implement one of many algorithms for solving a least-cost path because the model is based on a graph. There are a couple of different ways to approach solving the least-cost path: using an all-pairs shortest path algorithm or using a single-source shortest path algorithm. An all-pairs shortest path algorithm takes all nodes in the graph and finds the shortest path from each node to every other node. This can be useful when there are many moving objects In this case, all of the possible shortest path

calculations can be pre-computed and then loaded into the model so that the calculations do not need to be run more than once. One such implementation of an all-pairs shortest path algorithm is the Floyd-Warshall algorithm (Cormen *et al.*, 2001). The worse-case complexity of this algorithm is $O(n^3)$ and is suitable for small to moderate graphs (Cormen *et al.*, 2001). Since the Floyd-Warshall algorithm is only suitable for up to a moderate-sized graph, it is not suitable for this research. Even a small subset of the Salt Lake City road network around the University Line has over 5,500 nodes. With a worse-case processing time proportional to n^3, it may take an extremely long time to calculate the shortest paths for the network.

For the purposes of this research, it makes the most sense to use a single-source shortest path algorithm, which calculates the shortest path from a starting node (or set of nodes) to all nodes in the network. In the Salt Lake Valley (and almost all other real-world transportation networks), the network is sparsely connected, which means that street intersections (nodes in this model) are connected to only a small handful of other nodes when compared to the overall size of the rest of the network (Miller and Shaw, 2001). Since the network is sparsely connected, a single-source shortest path algorithm will be the most useful because it is computationally cheaper than an all-pairs shortest path algorithm. While the Floyd-Warhsall algorithm has a worst-case complexity time proportional to n^3, a single-source shortest path algorithm, such as Dijkstra's algorithm, has a worst-case complexity time proportional to n^2 (Worboys and Duckham, 2004; Cormen *et al.*, 2001). It is fairly intuitive why a single-source shortest path algorithm takes much less time to run than an all-pairs shortest path algorithm: there are drastically fewer shortest-path combinations that the algorithm needs to compute. For this model,

there is not just one bicyclist finding a shortest-path through network at one time. Since there are several origin nodes and several destination nodes that represent bicyclists moving simultaneously, the algorithm used in this model is a modified version of the Dijkstra algorithm that computes the least-cost path for each bicyclist before the cyclists start moving.

The Dijkstra algorithm, developed by Edsger Dijkstra, requires the edges (or arcs) to be weighted and non-negative (Dijkstra, 1959; Cormen *et al.*, 2001). The algorithm begins at the source node and assigns it a weight of 0, since it does not cost anything to reach the initial node, while also assigning the cost to reach all other nodes in the network a weight of infinity (or some very large number) (Worboys and Duckham, 2004). In various implementations of the Dijkstra algorithm, the weight or cost to reach each node can be derived from a variety of factors, such as distance, travel time, number of intermediate nodes or some combination. As the algorithm proceeds through the network, it updates the cost to reach each node by always choosing the least cost path to get to that node (Cormen *et al.*, 2001). The algorithm then iterates through the entire network until all nodes are reached, while updating the least-cost path at every iteration (Cormen *et al.*, 2001; Worboys and Duckham, 2004).

The implementation of the Dijkstra algorithm for this model is based on the *Being Kevin Bacon* model developed by William John Teahan (Teahan, 2010). The model uses the Dijkstra algorithm to measure the degrees of separation, or shortest-path, from each node to a designated center node (Teahan, 2010). The primary benefit of modifying this existing model is that the production time to create the model for this research is greatly reduced. There many other models created in NetLogo that implement various shortest-

path algorithms, such as the A* algorithm, which can be found either on the NetLogo *Community Models* webpage or by searching the web. Thus, the path-finding algorithm used in this thesis is largely based on the *Being Kevin Bacon* model with a few minor modifications to capture the proper arc weight. While the A* algorithm does usually perform more efficiently the than Dijkstra algorithm, the version of the A* algorithm used in an existing NetLogo model is a grid-based implementation (the agents move around in adjacent cells, or NetLogo patches). Since the *Being Kevin Bacon* model is a node-based model, it was much easier to implement for the purposes of this project.

The model used for this research is designed based on specific characteristics of the Salt Lake City area, and would need to be modified to apply to other areas. The model determines the least-cost path for agents to get from origin to destination using the Dijkstra algorithm. One of the primary challenges of computing cost in this model is to use distance based on those distances in the real-world, rather than using purely NetLogo space distances. Although the GIS extension in NetLogo does translate objects in GIS space into NetLogo space using a GIS map projection, the position of the objects is relative to other objects, rather than using map coordinates. The coordinates of any object in NetLogo represent the coordinates of the patch that object intersects. Thus, an intersection node located at 41˚10'31" N, 111˚25'52" W in the real world could have a NetLogo space coordinate of (-16, 10), indicating that it is 16 patches to the left and 10 patches above the origin patch (0, 0). The origin in NetLogo can either be the center of the NetLogo world, or it can be the patch in the lower-left corner. Consequently, the link-length function in NetLogo reports the length of any given link based on how long the link is in patch distances. This means that lengths must be converted to translate a

link-length of 3 into some measure of feet or meters represented on the ground. In the model this is represented by the link variable *beta*. The distance variable in the model is calculated as shown in Equation 3.1.

$$Distance = link\_length * beta \quad (3.1)$$

Distance is measured in feet, link-length is measured in the number of patches, and beta is a conversion factor. The reason for using the variable beta rather than just setting up a one-time conversion is that the value for beta (the conversion factor) will change based on the resolution (size) of the patches. One of the model settings allows the user to determine patch size to make the entire model larger or smaller. Thus, a link with length "3" at one resolution may have a length of "5" at another resolution, which means that the conversion factor must be adaptable to the resolution of the model. The beta variable is set up to translate the NetLogo distance into a real-world distance represented in feet.

3.5 Model Parameters

The primary variable that this model examines is *cost*. Cost is the weight assigned to each link (both streets and TRAX lines) for an agent (cyclist and TRAX trains) to traverse that link. The two primary factors involved in computing cost are the distance and the time it takes an agent to traverse each link based on its speed. It is plausible that a real-life cyclist would base his or her decision to choose either to use TRAX as part of the overall trip or just cycle the entire way on distance, total travel time,

or some combination of both. With this in mind, a portion of the cost calculation is allotted to the dummy variable *alpha* (α). The value of α has a discrete range of 0, 0.5 or 1. The value of α then indicates whether the agents are making their decisions to choose a route based on distance (finding the overall shortest path), time (finding the overall quickest path) or a combination of both in which distance and time are treated equally. For the purposes of this model, only time-based paths were considered in each experiment, so α was set to "1" at all times.

One of the other critical components in route choice for the cyclists is the roads available for travel. Although the road network in Salt Lake City is largely unchanged since the dataset was collected in 2006, there are several major roads in the city with high traffic counts which many cyclists prefer to avoid. Some of these roads, such as 400 South and Foothill Drive, are major thoroughfares to get people and goods to and from the university, whereas other roads, such as 700 East and State Street, primarily provide access to downtown and the fringes of downtown. A cyclist is legally entitled to ride on the side of the road unless posted otherwise. Still, legal entitlement does not translate directly into cyclists using roads where they do not feel safe. Thus, if major roads are disallowed for cyclist paths, the paths and behaviors of cyclists are likely to be closer to those observed in reality; however, if major roads are included in the path-finding calculations, it is possible to observe new and interesting behavior which may not be observable otherwise. The GIS data from the Utah AGRC do not explicitly have an attribute indicating whether a road segment is a major road or not. There is an attribute in the data indicating whether or not a road segment is either a Utah state highway or a United States highway. This attribute serves as a proxy for major roads in this model.

The inclusion or exclusion of major roads is based upon a binary on/off switch built into the user interface. The default setting for this switch is “on”, which means that all roads, major and otherwise, are included in the Dijkstra algorithm path calculations. When the users selects “off”, the major roads are hidden from view and assigned an arbitrarily high cost (1,999) so that major roads will not likely be part of a cyclist’s least-cost path unless her origin or destination is only reachable by a major road. For a map of the major roads used in this model, see Figure 3.1.

Another confounding factor in the cyclists’ decision-making is whether or not they choose to follow the posted rules on the TRAX vehicles. The rules explicitly state that there shall only be up to two bicycles at each end of each vehicle. This puts a strong limit on the number of cyclists who would be allowed to board any given TRAX train. The head-end of the first car is also not used by cyclists because this space is reserved for persons with disabilities. Thus, a train with only two cars (which is typical on the University Line) has room for no more than six cyclists: two at the rear of the first car and four in the second car.

In this model, the cyclists following the posted rules are represented as a percentage. A drop-down arrow in the user interface allows the user to select values of 0, 0.5 and 1. A value of 0 indicates that no cyclists are explicitly following the rules. Although it is possible for some cyclists to continue to follow the rules, this would be a coincidence and not an aberration. A value of 0.5 indicates that half of all of the cyclists will follow the rules and the other half will disregard them. A function in NetLogo allows half of the cyclists to be chosen at random each time the model is setup. Thus, those who follow the rules and those who choose not to will be different in each

simulation run. A value of 1 in the drop-down arrow indicates that all cyclists will strictly adhere to the posted rules. Thus, even if a cyclist arrives at a TRAX station and the TRAX-based path still represents the least-cost, the cyclist will not board the next train if it already is filled to capacity with other cyclists; it will re-evaluate the cost of waiting for the next train or continuing to his or her destination on a cycle-only path. The cyclists determine whether or not a train is full of cyclists by examining the *bike-cap* (bike-capacity) variable for the train. As a cyclist boards the train, the train reduces its bike-cap by 1; similarly, it increases its bike-cap anytime a cyclist de-trains. Each train has an initial bike-cap of 6, which presumes that all trains have two cars. Thus, a rule-following cyclist will not board a train with a bike-cap of 0 (or less), while a rule-disregarding cyclist will board any train traveling in the same direction the cyclist is traveling.

The behavior of the cyclists becomes more and more complex with an increasing number of other cyclists in the model. It would not be sensible to assume that each cyclist would start moving from origin to destination at the same time. With this in mind, each intersection node has a randomly assigned seed-time at the setup of each simulation run. The seed-time ranges from 0 to 3,600. Time in this model is measured in seconds, which implies that each cyclist will begin movement at some randomly assigned time within one hour of the simulation starting. During the execution of the Dijkstra algorithm, each cyclist determines a least-cost path through the network. The cyclist then determines when to move, based on the seed-time of the initial node and the time it takes the cyclist to traverse the street going from the initial node to the next node on her list of intermediate nodes. In NetLogo, time is represented in each model by a tick counter,

where one tick is one unit of time. For this model, one tick is equivalent to one second. The cyclist moves when the tick counter reaches the same value as the predetermined time to move. The time it takes a cyclist to traverse a street is determined by the speed of the cyclist.

The headway can also produce interesting ridership behavior. Typical headways on TRAX are 15 minutes on weekdays and 20 minutes on weekends. In this model, headway is modeled through the *train-headway* variable. Allowable values for this variable are 10, 15, and 20. While 15-minute and 20-minute headways are indicative of current service intervals, 10-minute headway is useful for representing a "what-if" scenario for policy-making. During the model setup, trains will spawn at different stations based on the selected headway. There are six trains moving concurrently under 10-minute headways, five trains with 15-minute headways and three trains with 20-minute headways. The starting stations for trains are loosely based on a 2010 TRAX schedule that was modified to meet the requirements of this model.

Another observed behavior in real-life is that some cyclists will not wait for the next TRAX train if they determine that they could get to their destination faster by continuing to cycle instead of waiting. While there is no known parameter explicitly stating how much time cyclists are willing to wait before starting to cycle again, this behavior is modeled in this thesis through *bike-wait-time* variable. In this model, bike-wait-time has discrete values of 0.6, 0.8 and 1, which represent a percentage of the train headway that the cyclists will wait. The cyclists calculate the maximum number of seconds if they had to wait for the train and then examine how many seconds are left until the next train arrives at that station. Each station (which is also an agent) keeps track of

when the last train was there and then starts counting down from the maximum headway time to the next train's arrival. This is analogous to the clocks at each TRAX station which display when the next train will arrive (in addition to the posted schedules). Although the rule-following behavior only applies to a percentage of cyclists, bike-wait-time applies to all cyclists equally. Thus, based on the allowable bike-wait-time variables, cyclists in this model will wait a maximum of 60%, 80% or 100% of the total train headway. If the time to the next train exceeds the maximum allowable wait-time, the cyclist proceeds to her final destination on a cycle-only path.

3.6 Rules Governing Agent Movement

One of the primary premises of agent-based modeling is that the rules governing agent movement and behavior should be fairly simple and straightforward (Flake, 1998). While complex behavior is an expected outcome of running an ABM, the complexity is derived from the agents interacting with their environment and other agents; it is not built-in based on a large set of complex rules. This section describes the specific choices each cyclist makes as it moves around in the model and implements the path-finding routine described in the previous section.

Each cyclist is spawned in the model at a random location. This initial location serves as the cyclist's origin for a trip through the model study area. Each cyclist is then assigned a destination node in the network that is also randomly selected. The set of destination nodes is randomly selected, and the destination for each specific cyclist is selected from the destination node list, which means that many instances arise where

more than one cyclist is pursuing the same destination. The specific means for choosing a number of cyclists during each simulation and the number of possible destinations is discussed in the following chapter.

Once a cyclist knows its origin and destination nodes, the first step is to run the version of the Dijkstra algorithm used in the model. The Dijkstra algorithm produces two outputs for each cyclist. The first array, known as *dijkstra-distances*, shows the cost for the cyclist to reach any node in the network from the origin node the cyclist is currently standing on. The second array, known as *dijkstra-directions*, shows the last node the cyclist would need to pass through to reach any node in the network. The cyclist compares the cost of a cycle-only path to destination with a cycle-only path to the nearest TRAX station. If the cost of reaching the nearest TRAX station is less than 50% of the cost of the cycle-only-to-destination path, the cyclist runs a procedure using the *dijkstra-directions* array to create a list of specific nodes to visit en route to the TRAX station. Otherwise, the cyclist creates a list of specific nodes to visit via a cycle-only-to-destination path. If the cyclist chooses a cycle-only path, it makes no further calculations and proceeds to the destination. If the cyclist chooses a path to TRAX, it moves to the nearest TRAX station before making further calculations. The nearest TRAX station for each node in the network was established by creating Theissen polygons in ArcGIS. Each node was assigned a closest TRAX station based on the polygon it intersected. These data were then brought into NetLogo through the GIS extension.

Once a cyclist reaches the TRAX station nearest its origin node, it then re-runs the Dijkstra procedure. It then evaluates whether or not it is cheaper to continue to destination via a cycle-only path or if a TRAX-based path still represents cost-savings. If

the cycle-only path is cheaper, the cyclist computes a path to the destination and leaves the TRAX station with no further calculations. If a TRAX-based path is cheaper, the cyclist then evaluates the time left before the next train arrives (using the bike-wait-time variable discussed in the previous section). If the time to wait exceeds the threshold established by the bike-wait-time variable, the cyclist computes a cycle-only path to destination. Since time is a major consideration for cyclist movement, it is assumed that the cyclist would abandon a cheaper TRAX path since it can probably save time (or perceive that it is saving time) by continuing to cycle rather than remain stationary at the TRAX station. If the time to wait is lower than the maximum threshold, the cyclist then determines which train is the one it will board when it arrives at the station. (Cyclists boarding TRAX need a specific train to board, since the next train to visit any given station could be traveling in the opposite direction of the cyclist's destination.) TRAX-riding cyclists then wait for their train to arrive and then ride the train to the station closest to their destination nodes. Of course, a TRAX-riding cyclist will know which station it must get off the train, since multiple cyclists are on the same train at the same time. Once a cyclist arrives at the TRAX station closest to its destination, it de-boards the train. Before any ensuing movement, the cyclist runs the Dijkstra procedure again and then computes a cycle-based path from the TRAX station to the destination node.

As each cyclist moves around in the model, it is keeping track of three things. The first item is the node it visits on all of the cycle-based segments of its journey. This list of nodes allows each cyclist's path to be reconstructed during the analysis of each simulation run. The second item is the time it visits each node. This parameter allows

the modeler to determine how long it takes each cyclist to complete its journey. The third parameter each cyclist retains is the distance of each link (edge) it traverses in the model. This is useful in making generalizations about the relationship between overall origin-destination distance and whether cyclists will travel to TRAX or use a cycle-only path. Once a cyclist reaches its destination the cyclist takes these three parameters and exports them as lists to an output window (that NetLogo exports to a text file). Since cyclist has accomplished its objective of reaching its destination, it then dies. Thus, as cyclists are moving around, the number of total cyclists is constant, which is to say that no cyclists in the model complete one trip and then re-spawn and make a return trip. As the simulation proceeds in NetLogo, the model stops running when all cyclists have completed their origin-destination trip. The set of decisions each cyclist makes is visualized as a flowchart depicted in Figure 3.2. While the set of choices a cyclist faces is far from exhaustive, the decisions represented here are some of the most likely choices each cyclist would have to make in a real-world setting. The model is dynamic in the sense that a cyclist choosing TRAX initially is not bound to that decision of it becomes apparent that a cycle-only path is cheaper at later time horizons. While a cyclist who chooses to cycle initially is bound to that decision, it seems reasonable that real-world cyclist would not choose to start cycling and then adjust her path to reach a TRAX station a later time horizons.

3.7 Conclusions

This section discusses the techniques used to create the ABM for this thesis. The first subsection outlines the study area used in the model and why that specific study area was appropriate. The second subsection discusses the various datasets used in the model and how those datasets needed to be modified to accommodate the needs of the ABM. The third subsection details how the model was constructed, including how real-world GIS data were imported into the NetLogo environment. The fourth subsection details the algorithm adopted for cyclist path-finding, including the modifications to the algorithm necessary to best fit the needs of the model. Subsection five discusses the parameters used in the model and the allowable values for each modeled variable. This is followed by a subsection regarding the rules governing cyclist movement, which is presented as a decision tree. In the following section, the ABM is tested through three distinct experiments that examine unique behavior based on model parameters.

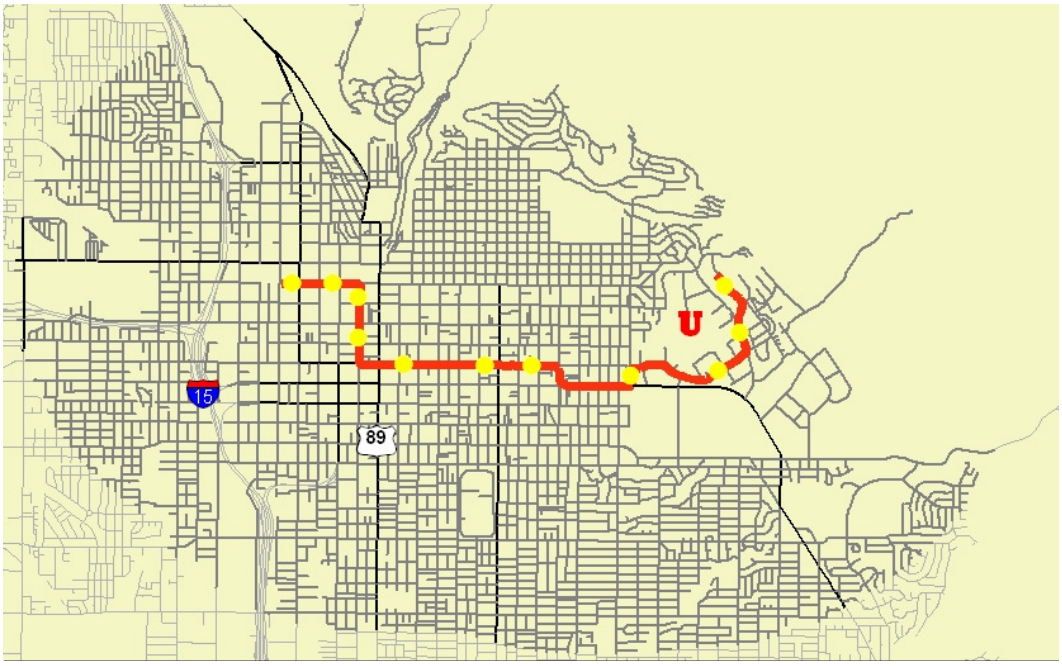

Figure 3.1 Study area for the ABM.

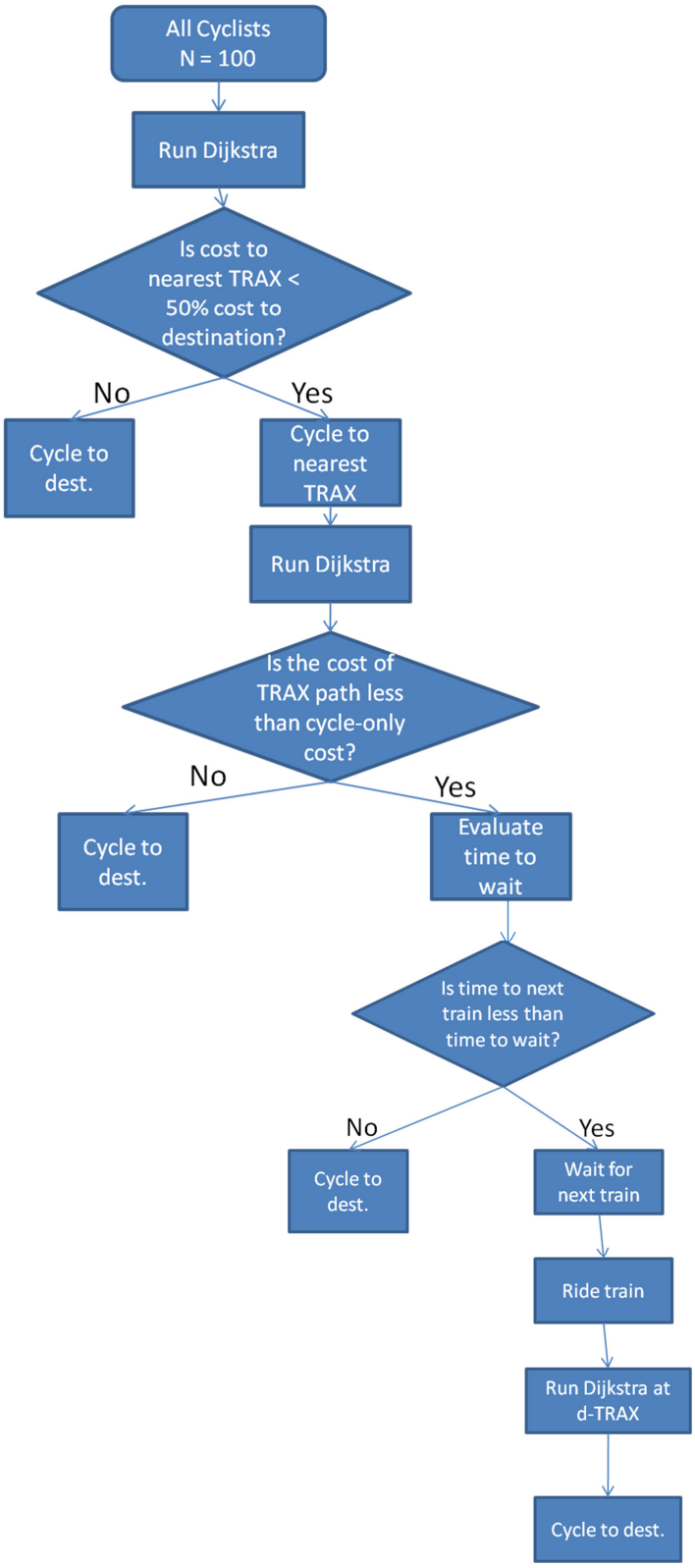

Figure 3.2 Rules governing agent movement.

Table 3.1 Cyclist speed based on street segment slope

Slope (s)	**Cyclist Speed**
>5%	5 miles per hour (440 feet / minute)
2 < s ≤ 5%	10 miles per hour (880 feet / minute)
-2 < s ≤ 2%	12 miles per hour (1,056 feet / minute)
-5 < s ≤ -2%	15 miles per hour (1,320 feet / minute)
≤ -5%	20 miles per hour (1,760 feet / minute)

4. RESULTS

This section shows the output of several experiments with the simulation model. These experiments were selected based on the model parameters described in the previous section. The first subsection describes the user interface in NetLogo. The second subsection follows this by describing the BehaviorSpace environment, which is the portion of NetLogo designed to run research-based experiments. The third subsection describes at length the output of all of the simulation runs. There were three groups of experiments, with a total of 19 individual experiments. Map and table-based outputs are shown to visualize and describe the differing model outputs based on the differing model parameters. Subsection 4.4 outlines a few simulations conducted against a very small real-world dataset provided by the Wasatch Front Regional Council. This section focuses on the actual model output, while analysis of the simulation outputs is reserved for Section 5.

4.1 Graphical User Interface

Since NetLogo is designed to be accessible to individuals with advanced programming skills as well as those with few programming skills, the user interface is also designed to be accessible to all types of computer users. It is quite simple for anyone to add buttons, choosers (drop-down menus), sliders and statistical plots. These model elements can be added to the map by choosing the "+" icon from the user interface. It is

possible to customize the model elements by right-clicking on them on then selecting Edit. Some model elements have required parameters (i.e., a statistical plot must have values for the x and y axes). Buttons are the most versatile model elements, since they can be programmed to run portions of code. The ABM used for this project uses model elements extensively, allowing the user to select the model parameters used in the simulation run. Figure 4.1 shows the home screen of the model as it appears when NetLogo initially opens.

This model is designed to begin loading input data as soon as the model is opened. This *startup* procedure is useful because it requires the GIS data to be loaded into the model only once, rather than at the beginning of each simulation. This procedure creates both the road network and the TRAX line network before any cyclists or trains are created. The elevation data also load during this procedure, and it displays in the background, or topologically underneath, the road and TRAX line networks.

The user prepares the model to run a simulation by pressing the Setup button. While the button could be named whatever the user wants it to be, it is common practice in NetLogo for almost all models to have a Setup button that calls the *setup* procedure from the model source code. It is also possible to run the *setup* procedure by typing "setup" into the NetLogo Command Center, located at the bottom of the screen. The setup procedure sets up the model based on the values of the model parameters. In this model, all of the model parameters are set using a drop-down menu. If the user changes the value of a model parameter after setting up a model, it is necessary to run the setup procedure again to reflect the new values for model parameters.

The model runs by pressing the Go button. The Go button is known as a *forever button*, since NetLogo will continue to loop through the *go* command until some criteria is met that stops the model. In this case, the model runs continuously until all of the cyclists have completed their journeys and have saved all of their list-based information to the output area. (See Section 4.3 for a discussion on the model output area.) The go procedure can also run from the Command Center, but typing "go" into the Command Center results in the procedure running only one time. Thus, a user only needs to click Setup followed by Go to successfully complete one simulation.

4.2 BehaviorSpace

NetLogo provides a native environment for running several simulations as part of an experiment, known as BehaviorSpace. The BehaviorSpace tool allows a NetLogo user to explore the parameter space within the model based on how one or more variables are set in relation to each other (Wilensky, 1999). This tool is very useful, particularly in situations where a sizable number of experiments need to run and it takes a significant amount of time (several minutes) to setup the model for one simulation. The experiments conducted as part of this research occurred in the BehaviorSpace environment (see Section 4.3 for a description of the experiments chosen for this research).

When a user selects the BehaviorSpace option from the Tools menu, the dialog that appears is designed to guide the user through the process of setting up an experiment. Figure 4.2 shows the BehaviorSpace experiment dialog window. BehaviorSpace can take one or many variables within the model to examine during an experiment. The user has

the option of giving variables discrete values (e.g., variable *x* can only have values of 10, 20 or 30) as well as a range of values (e.g., variable *y* can have values ranging from 10 to 100 with the intermediate values increasing by 10 each variation). The user needs to specify what command(s) are used to setup the model, run the model, and stop the model. There is also an option for Final Commands, which represent any commands NetLogo needs to execute after a model has stopped running before proceeding to the next experiment.

The BehaviorSpace environment is designed to give the user flexibility in terms of presenting the results of a simulation run. During an experiment, BehaviorSpace can monitor certain parameters, such as the number of cyclists who took TRAX as part of their trip or the mean distance that all cyclists traveled to complete their trip. NetLogo has the ability to write experiment results in spreadsheet format (as Comma Separated Value files) and in table format (as a text file). The model used in this project produced both a tabular and a spreadsheet output for each experiment.

BehaviorSpace has the ability to run multiple experiments simultaneously. The default setting is to run one experiment per processor core on the computer. While this would have been useful for this project, the simultaneous processing did not offer additional gains in terms of processing speed. NetLogo is setup to use up to 1 GB of available memory. It is possible to raise this memory ceiling, although doing so caused NetLogo to not initialize properly. Based on the way NetLogo handled simultaneous runs and an increased memory ceiling, it appears that during simultaneous simulation runs, each simulation is allocated a portion of the available memory, which in turn causes

the simulations to run slower than they do individually. Thus, all of the experiments conducted for this project only used one processor core.

In the NetLogo environment and in BehaviorSpace, the user can decide whether or not updates to the model are visible. Since updates to the model involve visualizing the agents as they move through the NetLogo world, turning the updates off causes the model to perform faster than it would otherwise. In this case, the agents are still moving around in the model and their movements are recorded, but the user is unaware of how quickly the agents are moving and how fast they are travelling. Thus, the BehaviorSpace environment allows the user to gain some performance enhancement over several simulation runs that would not be realized if each simulation ran one at a time.

For all of the experiments conducted during this project, the NetLogo model ran on the University of Utah College of Social and Behavioral Sciences TS1 terminal server. There were two primary advantages of using TS1. First, TS1 has more available memory than any of the individual workstations in the Department of Geography. Whether the model ran on TS1 or an individual workstation, it took several minutes to setup before a simulation would run. Although TS1 has more available memory, it did not necessarily process the model faster, since TS1 has many concurrent users all utilizing portions of the available memory. The second advantage of using TS1 is the ability to remotely connect and run experiments around the clock. While individual workstations could run the model a bit faster (setup time would be around 6 minutes instead of 8 minutes), the power settings on the individual workstations would not allow the model to run overnight if the user was no longer actively doing something on the machine. Thus, the ability to run

experiments on TS1 overnight and during the day allowed the entire set of simulations to get done more quickly than they would have otherwise.

4.3 Experiment Results

Three groups of experiments were conducted as a means of demonstrating the effectiveness of the model and meeting the stated objectives of this project. The three groups of experiments resulted in nineteen individual experiments with varying parameters. In the BehaviorSpace environment, the three groups of experiments were treated as the experiments themselves and then the simulation outputs were parsed into the individual experiments based on the model settings.

For each individual experiment, the target number of simulations was 35. The minimum goal for analysis was 20 simulations per experiment. If NetLogo encountered a memory-based error during a simulation run, the experiment stopped and needed to be restarted. The output of the simulations that completed before the error message was still saved, so the experiment did not need to restart from scratch. Some experiments were more prone to errors than others, which is likely a result of some parameters causing processing time to be substantially more than other parameter settings.

The three groups of experiments are train headway versus bicyclist wait time at TRAX stations, train headway versus whether or not cyclists follow the UTA rules, and considerations of elevation (on / off) versus considerations of major roads in path-finding (on / off). These experiments were selected for analysis because the parameters measured present a unique opportunity to showcase the benefits of using agent-based

modeling for this type of application. The following subsections present the outputs of each experiment. The discussion on each experiment includes the number of successful simulations per experiment as well as a series of maps and tables showing cyclist behavior under the experimental conditions.

4.3.1 Experiment Set One

The first set of experiments examines cyclist behavior based on the inclusion of elevation data and whether or not major roads are allowable for cyclists to use. The elevation variable is particularly crucial to this project, namely to Objective 1 which seeks to demonstrate that an ABM is appropriate tool for analyzing a mode-choice problem. In this set of experiments, the elevation is of each node is calculated based on the grid cell value in the LiDAR data, as described in Chapter 3. The elevation data are treated in NetLogo as a binary (on / off) variable.

The major roads variable attempts to examine some level of perceived cyclist safety. These experiments assume that cyclist behavior will be different if they are allowed to use roads that generally have higher motor vehicle speeds and higher traffic volumes than it would be if cyclists are attempting to avoid such streets for their own personal safety. Table 4.1 shows the number of successful simulations per experiment. Each cell in the 2 x 2 matrix represents one individual experiment. Figure 4.3 presents the map output of the first experiment with elevation turned off (all nodes are assigned an elevation of 1 and streets a slope of 0) and major roads turned off. Figure 4.4 shows the map output of the fourth experiment with elevation turned on and major roads turned on.

Figure 4.5 shows a histogram of the percentage of cyclists who took TRAX in each experiment in the first set of experiments. Figure 4.6 shows the grand mean and standard deviation of cyclist trip distances in each experiment. Table 4.2 presents summary statistics for each experiment, including the grand mean and standard deviation of cyclist trip distances as well as total overall number of cyclists who took TRAX and the total number of cyclists who choose to cycle for their entire trip. The results shown on the following maps represent the number of times a particular road segment was visited, or traversed, by a cyclist. While the TRAX lines are depicted on the maps, they are not drawn based on the number of times a cyclist rode a train. (This type of visualization would cause difficulty determining the number of times cyclists rode on the streets TRAX runs, namely South Temple, Main Street, and 400 South.)

Based on the above maps, charts, and table, there are a few noticeable patterns in the output data. First, in Figure 4.3, there are very few trips (sometimes none at all) on the major roads. Since the major roads were turned off for shortest path calculations, this is expected. With this is mind, there are a relatively high number of trips on 300 South, which is one block north of the main portion of the east-west section of the TRAX line. In Figure 4.4, it is evident that cyclists will use major roads if those roads are allowed, since the segment of Foothill Drive near the Research Park area of the University was visited over 200 times. Also, the inclusion of elevation in the path-finding routine appears to cause a significant number of downhill (westbound) trips along South Temple and 100 South.

In Figure 4.5, there is a noticeable jump in the overall percentage of cyclists taking TRAX in experiments 3 and 4, which represent experiments with elevation on.

Although experiment 4 had the highest overall number of cyclists using TRAX (as shown in Table 4.2), experiment 3 had the highest percentage of cyclists taking TRAX at 16.86%. Part of the reason that experiment 4 had a higher overall number of cyclists using TRAX is that there were more successful simulations producing usable output than there were in experiment 3. Although the varied model parameters did seem to have an effect on the number of cyclists taking TRAX, it did not seem to impact the overall trip distance. In the four experiments, the range of trip distances was 3.94 miles to 4.06 miles and the standard deviation of trip distances ranged from 1.99 miles to 2.06 miles.

4.3.2 Experiment Set Two

The second set of experiments examines cyclist behavior based on the frequency of TRAX trains and whether or not the cyclists observe the posted rules on the trains regarding the bicycle capacity in each car. The values of train headway represent two actual observable train headways and one hypothetical value. During the weekdays, all TRAX trains run at 15-minute intervals. On weekends, trains on both lines run at 20-minute intervals. In this experiment, trains can also run at 10-minute intervals. Thus, the individual experiments with trains running at 10-minute headways represent the implications of a potential policy decision for UTA if economic conditions and TRAX ridership increases from current levels.

The rules variable is derived from real-world observation. The posted UTA rules state that only two bicycles are allowed at each end of each car at any given time. As noted in Chapter 3, the rules variable can have a significant impact on simulation

outcomes if all of the cyclists are following the rules and some cyclists are forced to continue their trip by getting back on their bicyclist to finish their trip. Table 4.3 shows the number of successful simulations per experiment. Each cell in the 2 x 3 matrix represents one individual experiment. Figure 4.7 shows the map output of the seventh experiment with train headway set to 15 minutes and cyclists not following the rules. Figure 4.8 represents the map output of the tenth experiment with train headway set to 20 minutes and cyclists following the rules. Figure 4.9 shows a histogram of the percentage of cyclists who took TRAX during each experiment in the second set of experiments. Figure 4.10 shows a histogram of the grand mean of cyclist trip distances and the standard deviation of cyclist trip distances. Table 4.4 is a summary of all of the model parameters and the outputs during each of the six experiments in the second set. The maps selected for the second set of experiments represent the maps that have the most contrasting output for ease of comparison. The maps show the number of times a road segment was visited during the simulation runs for the individual experiment displayed.

Several trends are evident based on the output results of the second set of experiments. First, in Figure 4.7, it is evident that certain street corridors were heavily used by the cyclists, while others were not. As one travels from east to west, 200 South was frequently used as far west as 700 East, where a large contingent of trips traveled north to 100 South and then continued west on 100 South to State Street. Also in Figure 4.7, there are large arrows leading away from the Arena and Temple Square TRAX stations, which suggest that several cyclists arrived at the station and determined that the cost to continue cycling was lower than waiting for the train. In this set of experiments, major roads were allowed in the path-finding routine. This is evident in the large number

of trips headed north on Foothill Drive in the direction of the South Campus TRAX station. Figure 4.8 shows the output of experiment 10, which shows the output for experiments with long train headways (20 minutes) and rule-following cyclists. In this experiment, there are fewer well-traveled corridors. South Temple shows a high number of trips, mainly headed eastbound. The 200 South to 700 East to 100 South corridors is somewhat visible, but not nearly as much as it was in Figure 4.7.

Another trend in this set of experiments is that the experiment with the highest overall number of cyclists who took TRAX is also the experiment with the highest percentage of cyclists taking TRAX. Figure 4.9 shows that experiment 5 had the highest percentage of cyclists taking TRAX. This experiment represents 10 minute train headways and cyclists not following the posted UTA rules. Figure 4.9 shows that experiment 7 had the lowest percentage of cyclists taking TRAX. This result is counterintuitive because the headway (15 minutes) is not very high and the cyclists are not following the rules.

Table 4.4 shows the grand mean of cyclist trip distances and the standard deviation of cyclist trip distances. This table shows results similar to those of the first set of experiments. The range of mean trip distances is 0.15 miles (3.91 miles to 4.06 miles), and the standard deviation of trip distances ranges from 1.97 miles to 2.03 miles. The range of 0.15 miles is just slightly longer than a Salt Lake City block (roughly 0.14 miles).

4.3.3 Experiment Set Three

The third set of experiments examines cyclist behavior based on varying the headway of the TRAX trains and the amount of time a cyclist is willing to wait at a TRAX station. The cyclist wait time parameter was not included in the WFRC transit survey data. Thus, the values for the wait time parameter represent hypothetical discrete values, while any such parameter may be continuous in reality. In this set of experiments, the allowable wait time values are 0.6, 0.8 and 1, where 0.6 represents cyclists waiting at the TRAX station for 60% of the total train headway, 0.8 represents an 80% of headway wait threshold, and 1 represents cyclists who will wait for the next train regardless of how long it takes for that train to arrive at the station. The parameters from train headway are the same as in the second set of experiments. Allowable headway values are 10 minutes, 15 minutes and 20 minutes.

Table 4.5 shows the number of successful simulations per experiment. Each cell in the 3 x 3 matrix represents one individual experiment. In each of the individual experiments, NetLogo encountered memory-based run-time errors. Under normal operating circumstances, NetLogo typically returns the specific identification number, or *who number*, of an agent that confronts an error in the model code. During all experiments, NetLogo confronted several errors that ascribed that error to “person -1”. Since all of the who numbers start at 0 and increase at each simulation run, it was unclear exactly which specific agent was attempting to run the erroneous bit of code. Additionally, NetLogo did not reference which specific line of the code caused the error. (Typically, an error message will highlight the specific portion of the code that generates the error message.) The NetLogo documentation did not offer guidance directly for this

sort of error message, but the size of the model attempting to handle all of the GIS data is one potential explanation. The documentation states that NetLogo has been tested with models of moderate size, but not very large models (Wilensky, 1999). Also, since the GIS extension to NetLogo has only been under development for the last couple of years, there could be several bugs with such a large NetLogo world in this model.

Figure 4.11 shows the map output of experiment twelve with the headway set at 10 minutes and the wait time set at 0.8. Figure 4.12 shows the map output of experiment sixteen with the headway set at 15 minutes and the wait time set at 1. Figure 4.13 shows the map output of experiment nineteen with the headway set at 20 minutes and the wait time set at 1. Figure 4.14 shows a histogram representing the percentage of cyclists who took TRAX during their trip. Figure 4.15 shows a histogram of the grand mean of cyclist trip distances and the standard deviation of cyclist trip distances. Table 4.6 depicts the outputs of the experiments based on the given model parameters.

The output of the third set of experiments reveals a couple of trends observed in previous experiments, as well as a few previously unobserved patterns. In Figures 4.11, 4.12 and 4.13, the South Temple corridor between the Arena TRAX station (300 West) and 700 East is heavily traveled. This trend is similar to those in previous experiments. As in previous experiments, many westbound trips from the University toward downtown used the 200 South to 700 East to 100 South corridors. All three maps also reveal a clear trend of trips traveling north on Foothill Drive to Mario Capecchi Drive leading toward the South Campus TRAX station.

In contrast to previous experiments, Figures 4.11, 4.12 and 4.13 show a clear pattern of trips lead to the Arena TRAX station coming from the north on 300 West. One

of the more peculiar trends in all three figures is that many southbound trips around the University use 1100 East to 600 South to 1200 East to 800 South. Since 1300 East is a major thoroughfare in that area and has higher speeds and more lanes, this trend is counterintuitive from a regular user's point of view. All three figures also show a large number of trips leading north away from the Stadium TRAX station before accessing 200 South.

Figure 4.14 depicts an intriguing pattern different from all of the simulation outputs. This histogram clearly shows the three different values for wait time at the TRAX station (0.6, 0.8 and 1) and the percentage of cyclists using TRAX increasing as wait time increases. In experiment 13, with a train headway of 10 minutes and a wait time of 1.0 (100% of total headway), an astounding 20% of cyclists used TRAX. This value represents the highest percentage of TRAX users in any of the experiments in this project. As shown in Table 4.6, there were over 600 cyclists using TRAX in experiment 13. This experiment was also the only experiment that exceeded 600 cyclists using TRAX.

4.4 Survey Data Comparison Tests

The Wasatch Front Regional Council provided data from a 2006 comprehensive transit user survey conducted by NuStats. While the number of usable bicyclists in the survey is small compared to the overall number of survey responses, this dataset does provide some insight into how well the parameters in this model perform against real world data. As noted in Section 3, NetLogo randomly generated the origin location of all

cyclists in this model. The destinations locations were also randomly generated, although NetLogo only generated 50 destination locations for every 100 origin locations (in hopes of getting the model to process faster). The NuStats data offers a different perspective for model performance because the origin and destination locations of the cyclists are fixed for each simulation.

The NuStats survey covered transit users of all types of transit in the UTA system in 2006. Survey administers rode buses and TRAX trains and handed out the survey to patrons after they boarded the vehicle. The survey itself was designed to be filled out in a very short period of time given that some patrons may only be on transit for a short time. There were 4,726 responses within the survey dataset. In addition to the wealth of demographic data collected in the survey, the survey noted origin location, destination location, transit access mode and transit egress mode.

The number of usable responses amounted to those responses which involved a complete origin-destination trip wholly contained within the study area, used a bicycle as the transit access mode, and used the University Line on the trip. There were 120 survey responses representing transit users who used a bicycle as the transit access mode. Of these 120 records, only 13 represented trips that were wholly contained in the study area and used the University Line. (These trips also did not use the Salt Lake City – Sandy Line, even though a small portion of this line falls within the study area.) Thirteen responses are too few to constitute a statistically large sample, but it is useful to draw some preliminary conclusions about the effectiveness of the model. Since the origins and destinations of the cyclists are fixed from one simulation to the next, the shortest path will likely be the same for most simulations. Cyclist behavior can still vary even if the

shortest path is the same or very similar multiple times based on the other variables in the model.

A series of experiments were conducted using the WFRC data as a means of testing the effectiveness of the model. The model was modified slightly from its state in Section 4.3 to accommodate the fixed locations of cyclist origins and destinations. The only component that needed to be changed was the method for generating where the cyclists spawned and how they selected their destinations. The path finding routine and all other components of the ABM remained unchanged. Since the number of usable cyclists in the dataset is so small, the WFRC data was tested using only 20 simulations per experiment. Testing involved the same 19 individual experiments as in Section 4.3. The dataset shows real-world examples of actual TRAX users, which implies that the model parameters that cause the highest number of cyclists to use TRAX in the model are those parameters that most closely resemble actual cyclist decision-rules. In all experiments, all 20 simulations ran successfully without encountering any of the errors mentioned in Section 4.3. Figure 4.16 shows a map of the origin and destination locations of the 13 cyclists in the WFRC dataset.

The test of the WFRC data produced the same output as those in the previous section, except that the output was not mapped. The output of the simulations with the WFRC data was unexpected. For the first set of experiments, which test the effects of elevation against the effects of major roads, a total of 80 simulations were conducted (4 individual experiments ran 20 times each). Of these 80 simulations, there were a total of 1,040 cyclists, and only 69 took TRAX. In all individual experiments, there were no

experiments with more than one cyclist who took TRAX, and there were several experiments where no cyclists took TRAX.

In the second set of experiments, which tests the effects of train headway against the effects of cyclists following the rules, a total of 120 simulations were conducted (6 individual experiments ran 20 times each). As with the previous set of experiments, the maximum number of TRAX riders was one for any given simulation. There were 1,547 cyclists represented in these experiments, and only 107 took TRAX. While there were some simulations where zero cyclists took TRAX, this occurred at all parameter settings, which does not suggest a clear trend as to what would cause the one TRAX rider to finish the trip by cycling only.

The third set of experiments tests the effects of train headway against the effects of cyclist wait time at TRAX stations. In this set of experiments, there were a total of 180 simulations (9 experiments with 20 simulations each). The maximum number of TRAX riders per simulation was one. Of the 2,325 cyclists represented in all of the simulations, only 133 took TRAX. Some simulations had no TRAX riders at all, but it was not clear that this behavior was more likely under any given set of model parameters than it was under others. Under many circumstances, the sum of the cyclist distance traveled and the standard deviation of the cyclist distance traveled were the same for multiple simulations. While these values did change slightly, it was common for the majority of simulations in each experiment to have the exact same reported sum distance and standard deviation of distance values. The low number of cyclists taking TRAX in the WFRC tests seems to imply that there were other confounding factors influencing agent behavior that were not included in this model.

4.5 Conclusion

This section details the outputs of the simulation runs from the ABM. The first subsection shows the model interface in NetLogo and the options available to the user upon startup. The second subsection describes the BehaviorSpace environment, which is the native environment in NetLogo to setup experiments for research applications. This section also noted the computing environment used to run the simulations. The third subsection detailed at length the output of the experimental simulation runs. This subsection included maps and tables for each individual experiment, showing the differing model outputs based on differing parameters. The fourth subsection of the chapter described some brief simulations conducted against a very small dataset provided by the Wasatch Front Regional Council. The output of these simulations was rather unexpected. The next section offers analysis of the results, including what the results imply about the ABM environment, the model parameters, as well as the use of ABM for multimodal travel models.

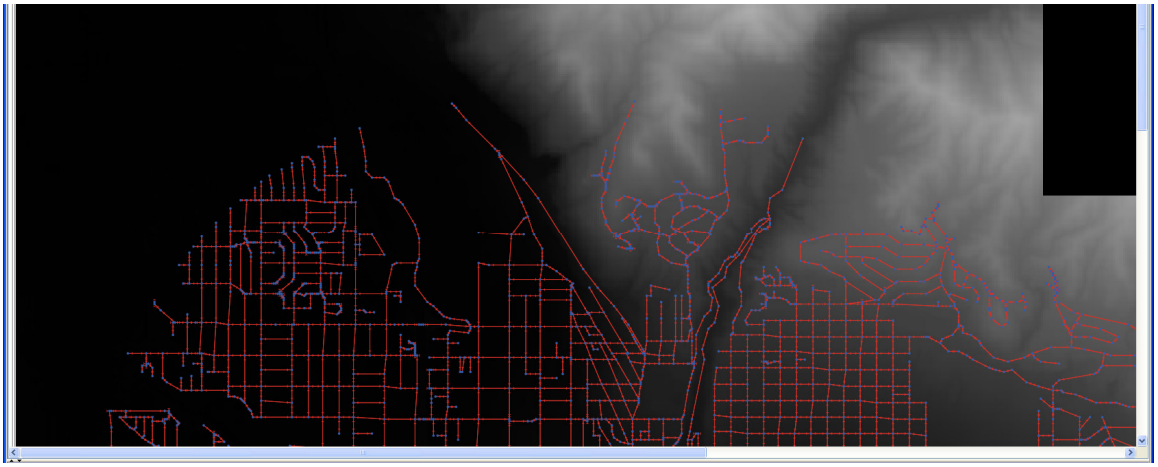

Figure 4.1 The NetLogo world after the model initializes.

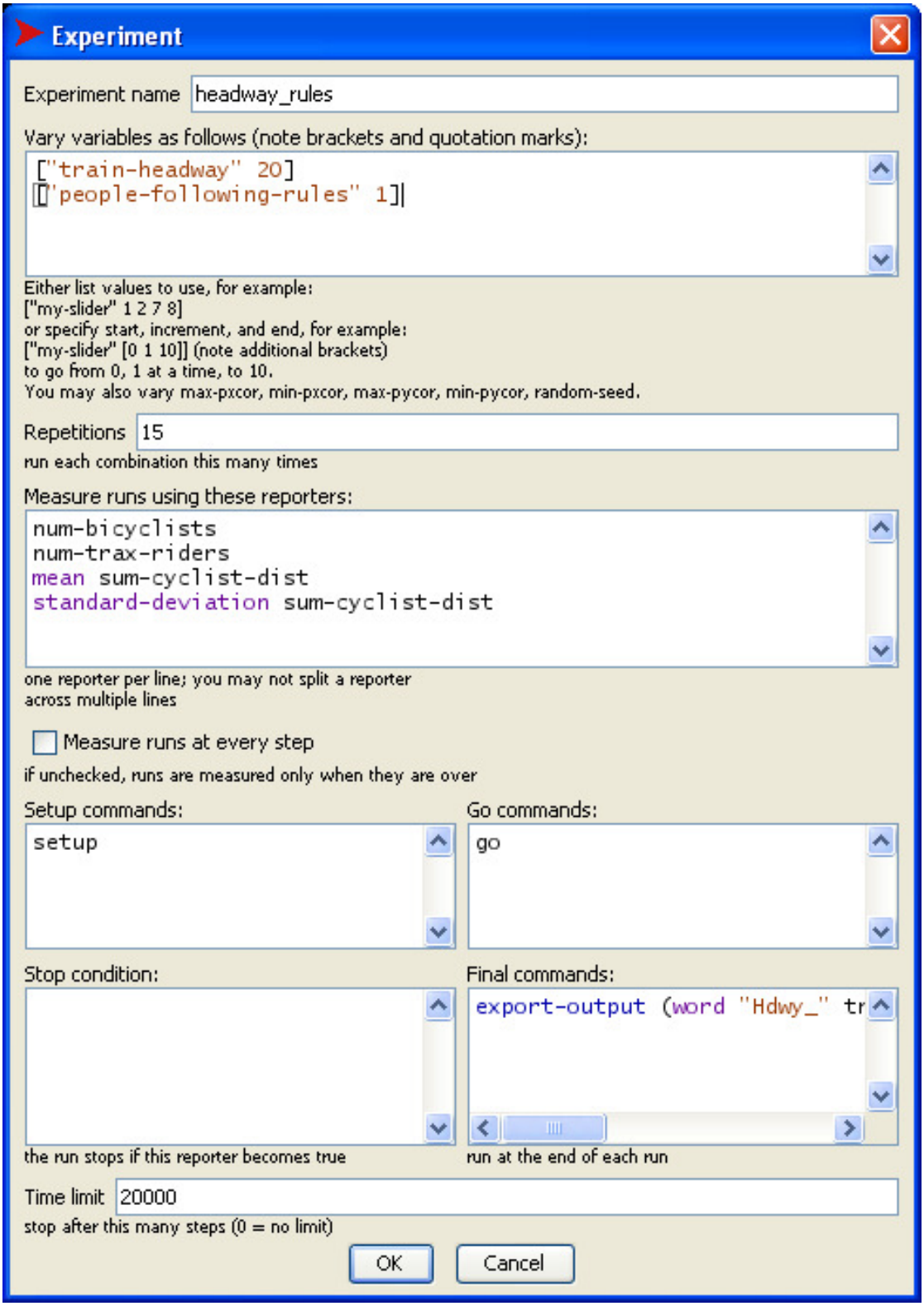

Figure 4.2 The BehaviorSpace dialog menu.

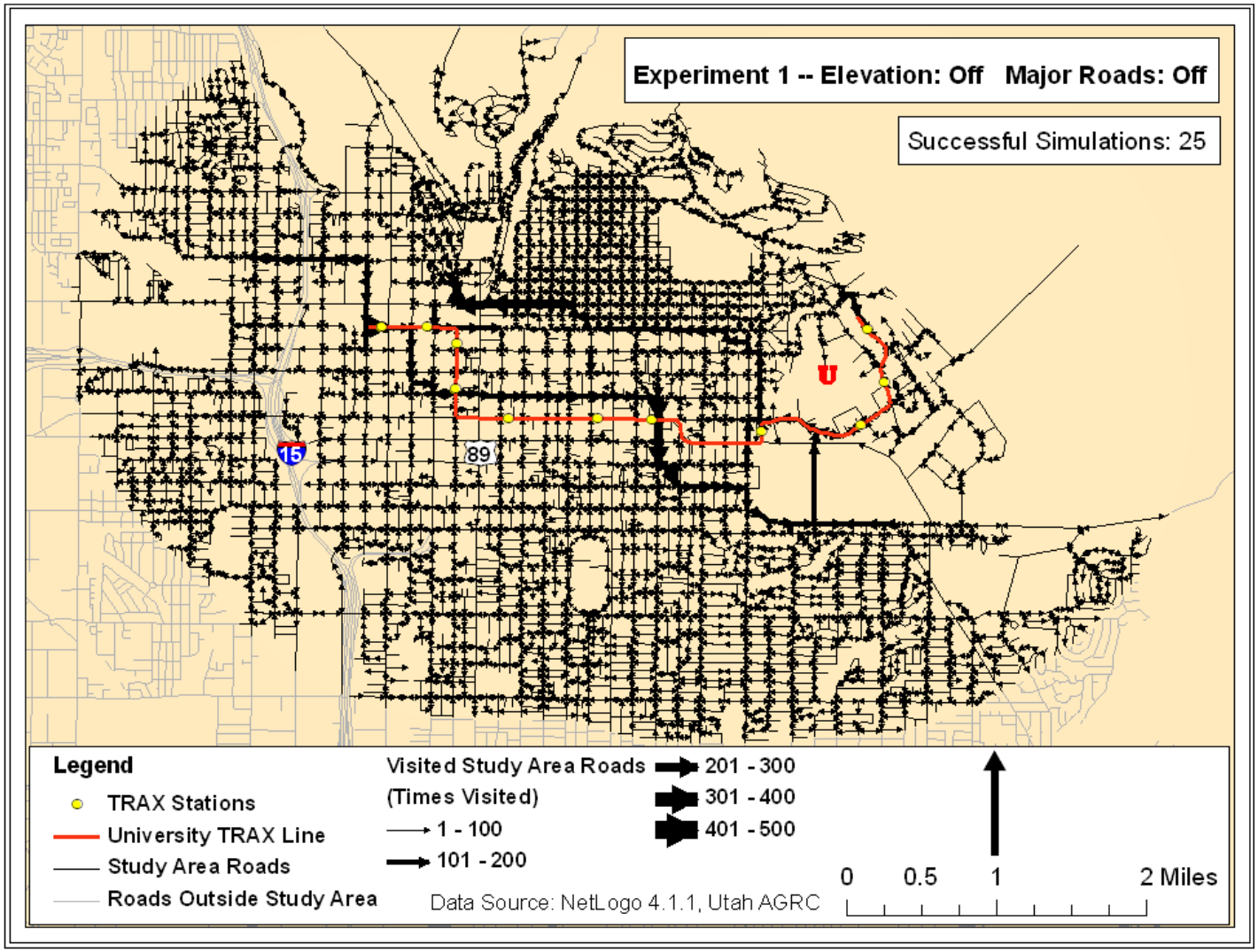

Figure 4. 3 Map output of experiment 1: elevation = off and major roads = off

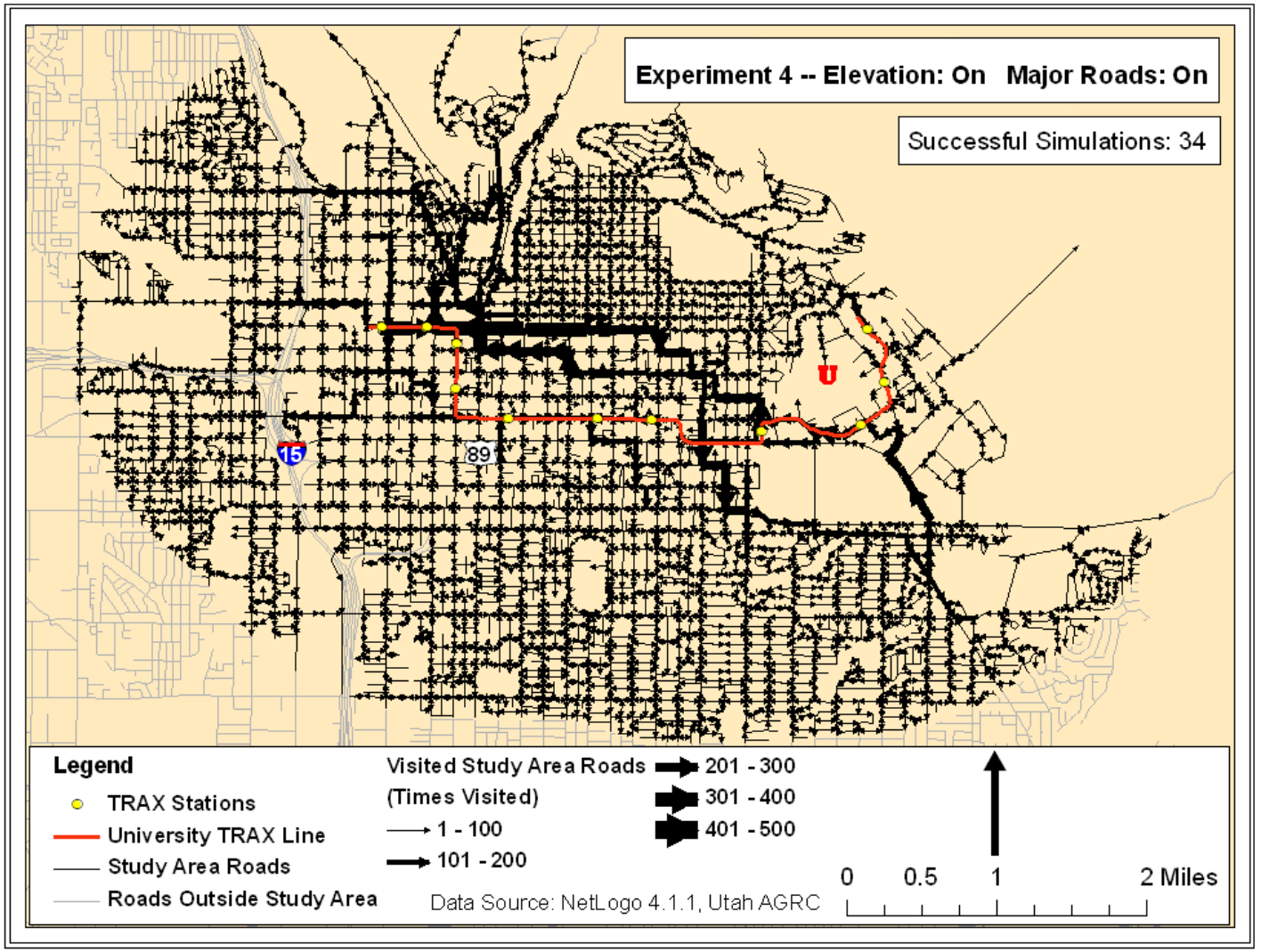

Figure 4.4 Map output of experiment 4: elevation = on and major roads = on.

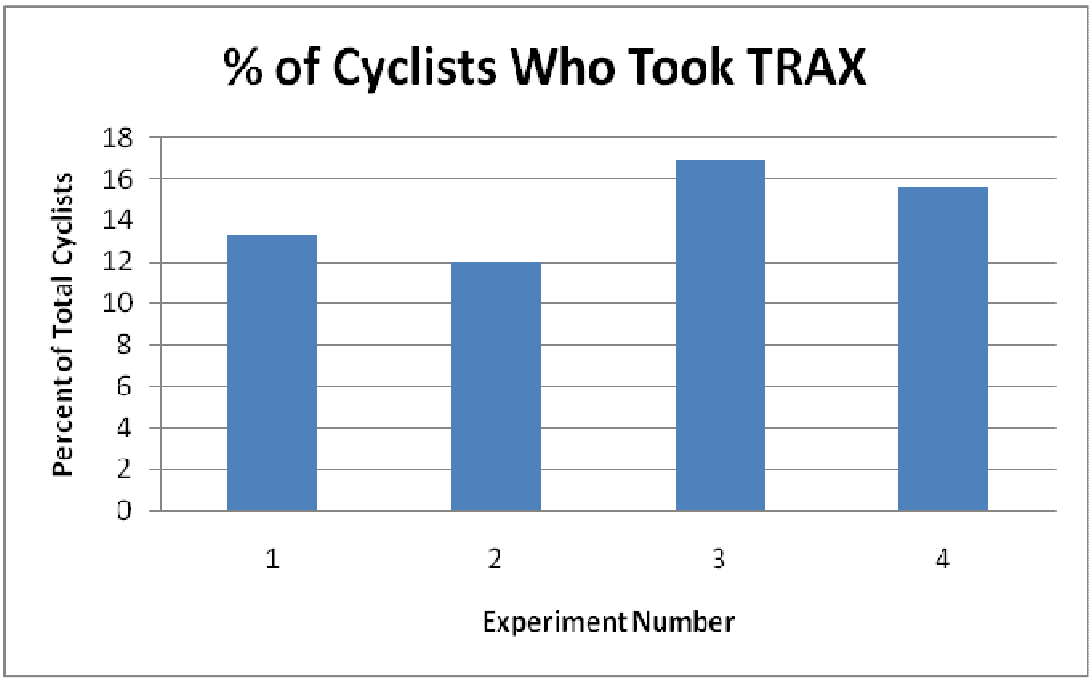

Figure 4.5 Histogram of the percent of cyclists who took TRAX in the first set.

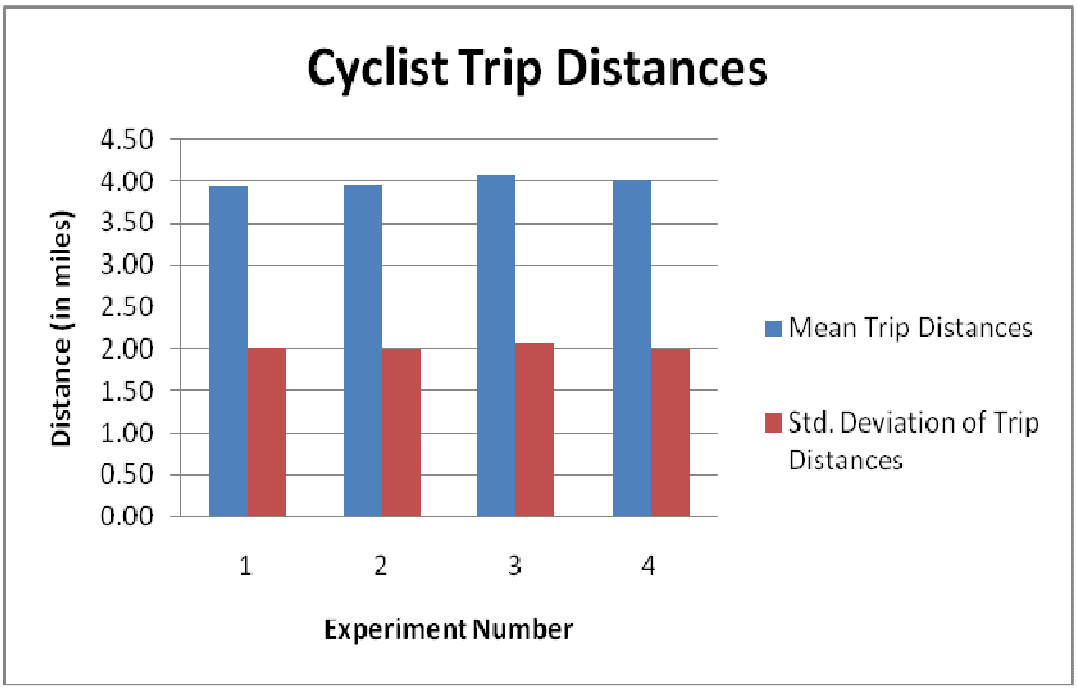

Figure 4.6 Histogram of the mean cyclist trip distance in the first set.

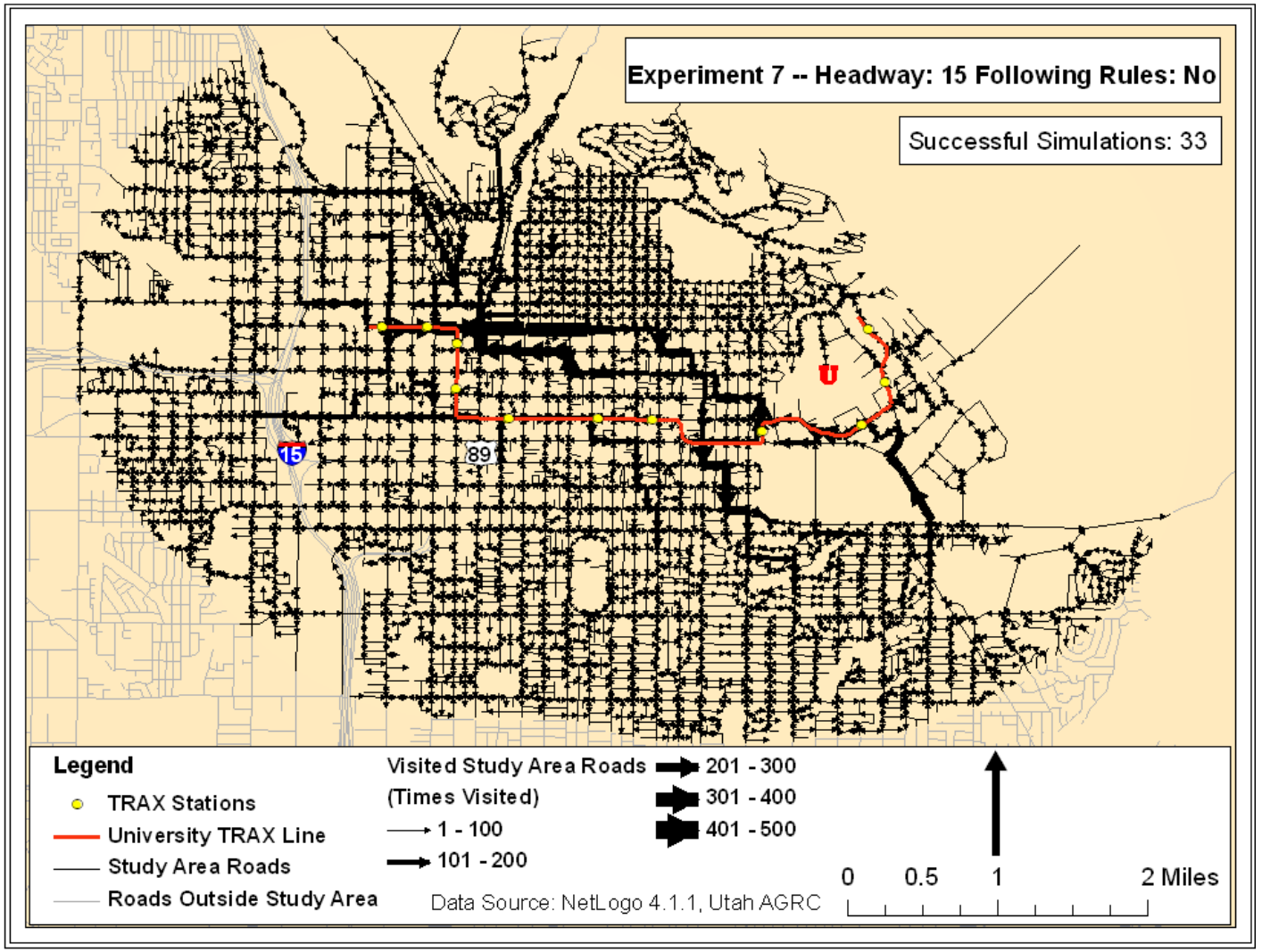

Figure 4.7 Map output of experiment 7: headway = 15 minutes and rules not observed.

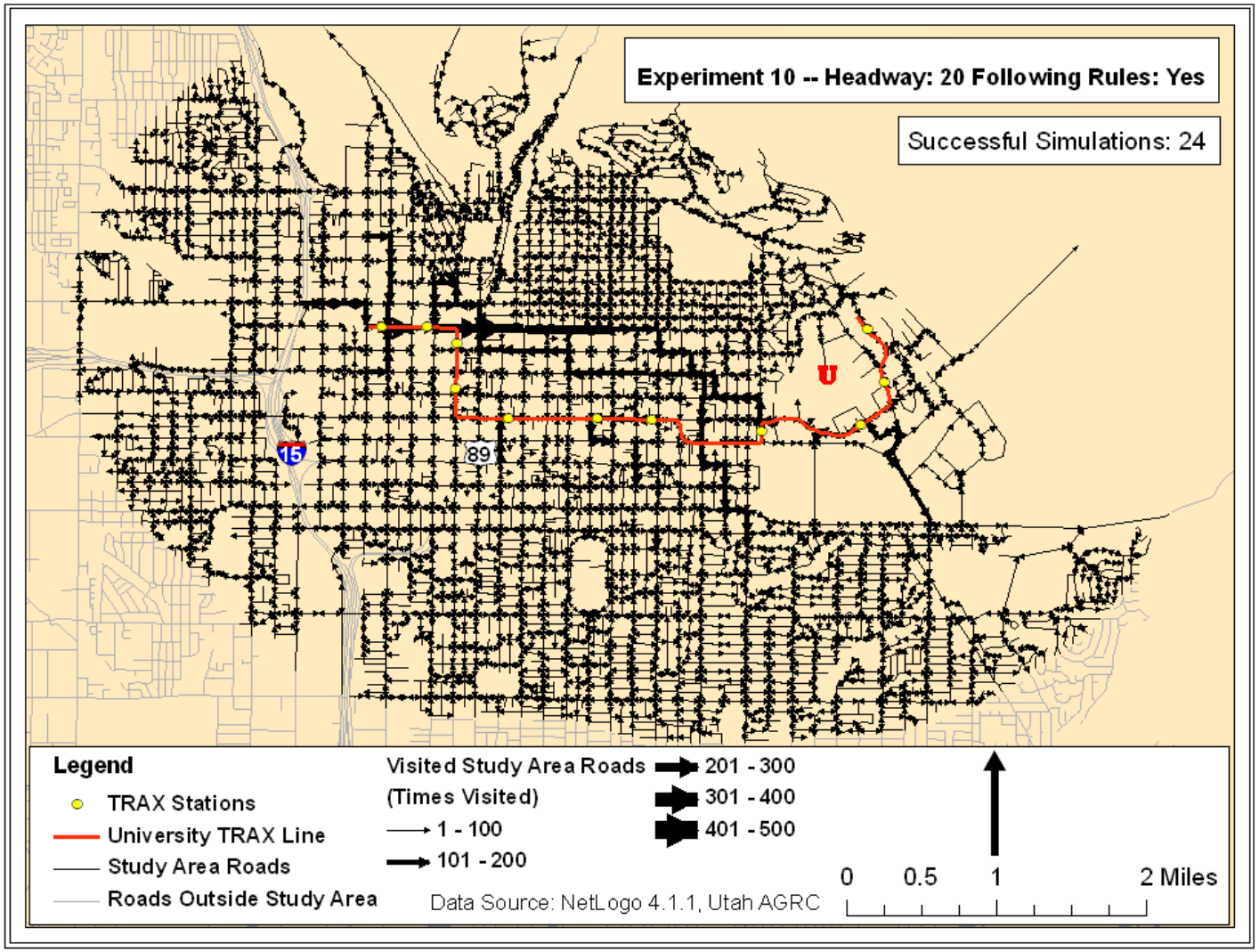

Figure 4.8 Map output of experiment 10: headway = 20 minutes and rules observed.

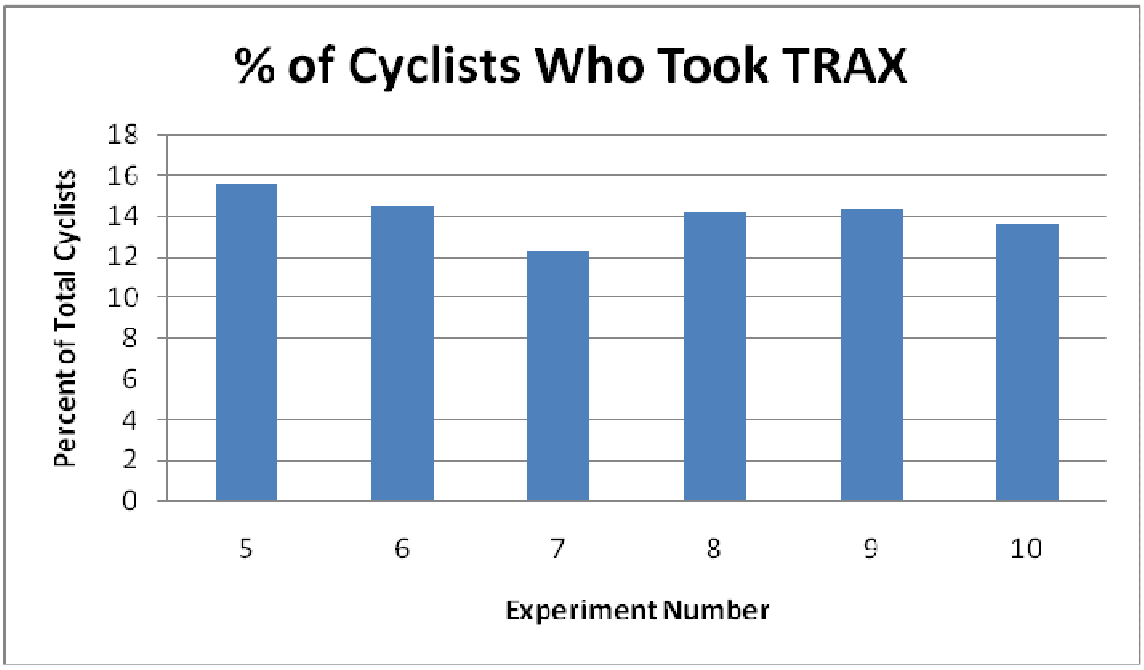

Figure 4.9 Histogram of cyclists who took TRAX in the second set.

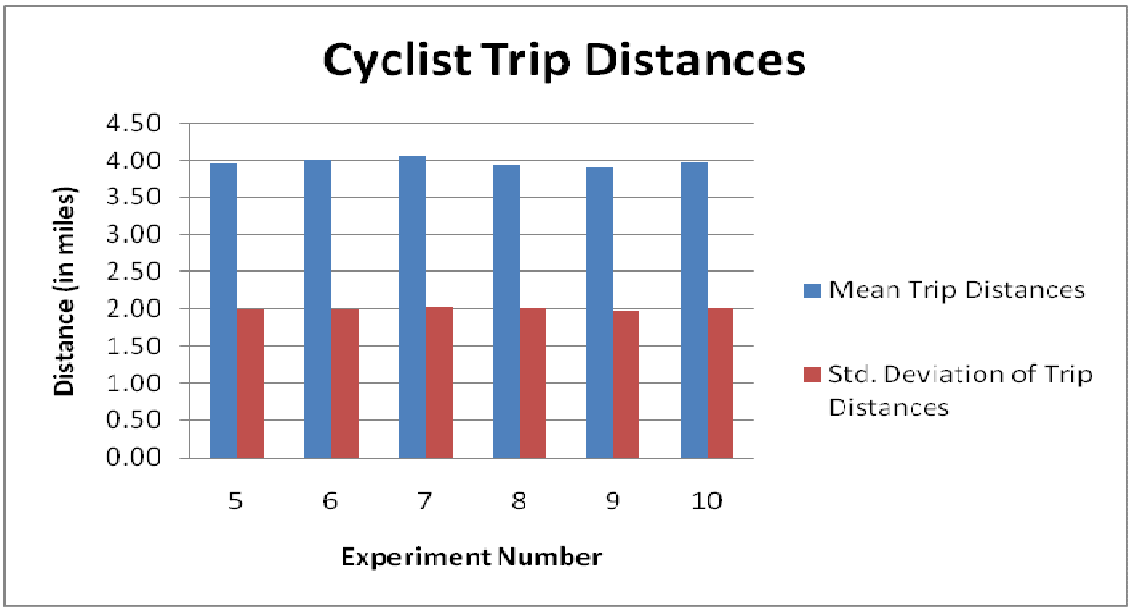

Figure 4.10 Histogram of the mean cyclist trip distances in the second set.

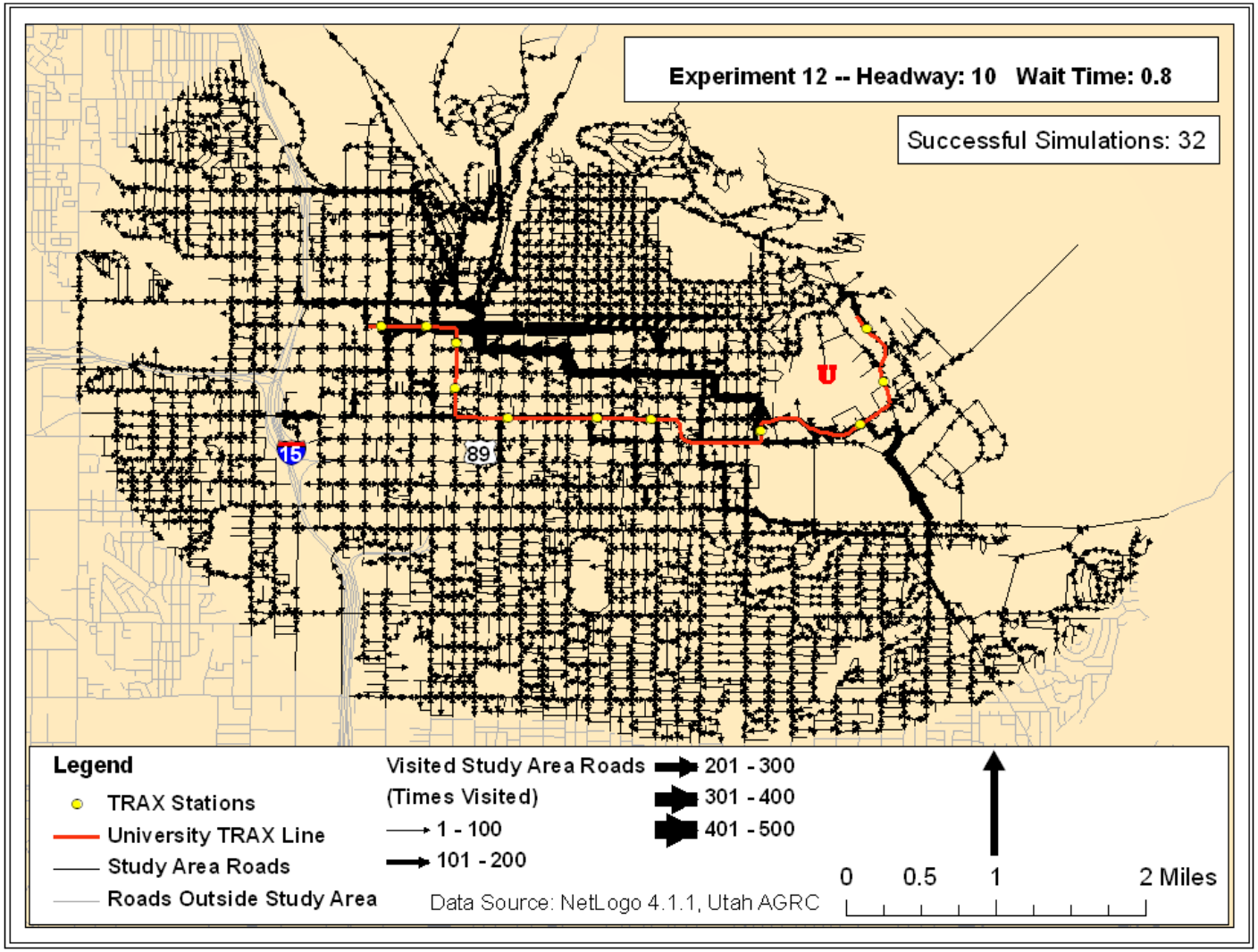

Figure 4.11 Map output of experiment 12: headway = 10 minutes and wait time = 0.8.

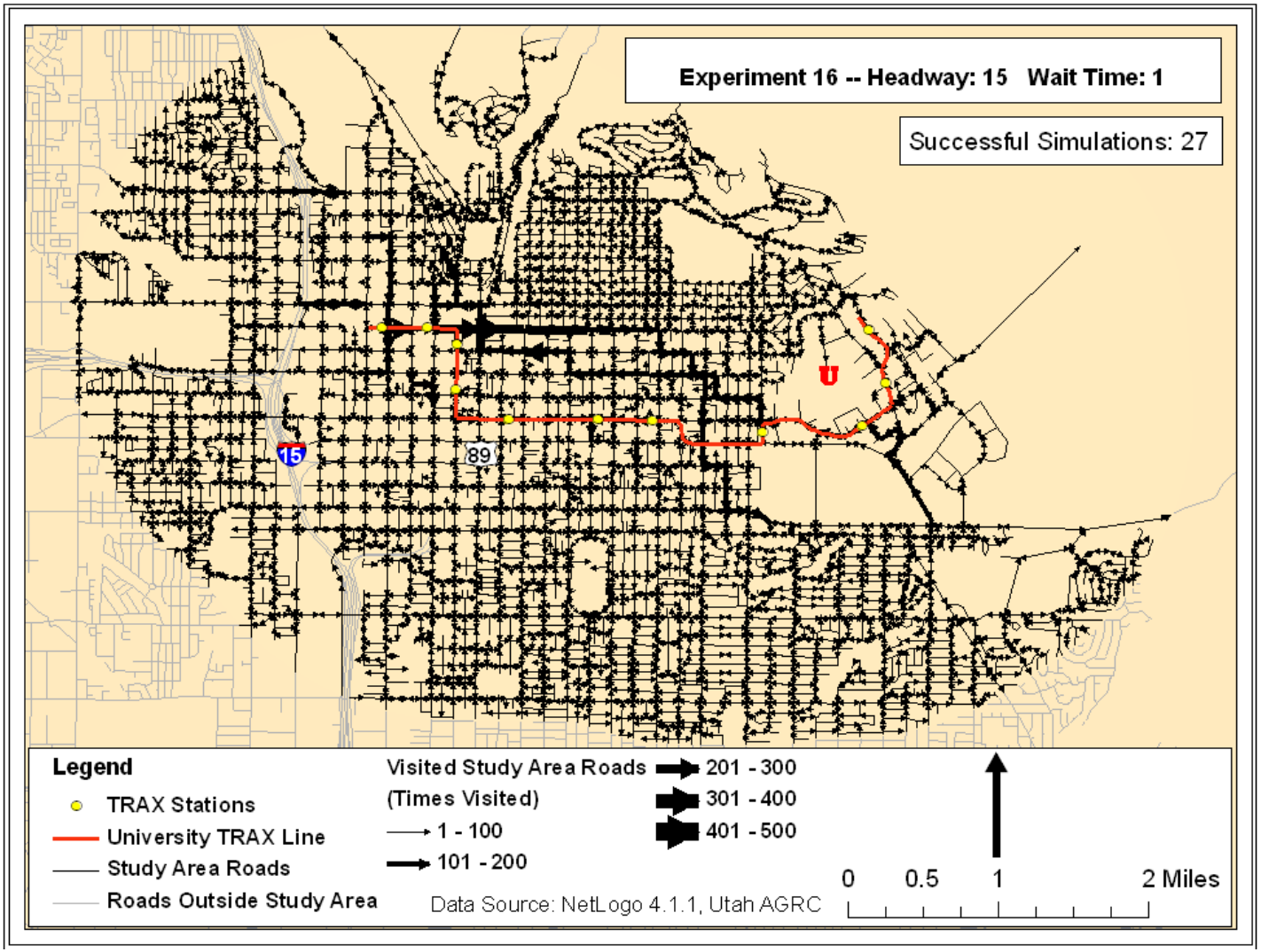

Figure 4.12 Map output of experiment 16: headway = 15 minutes and wait time = 1.

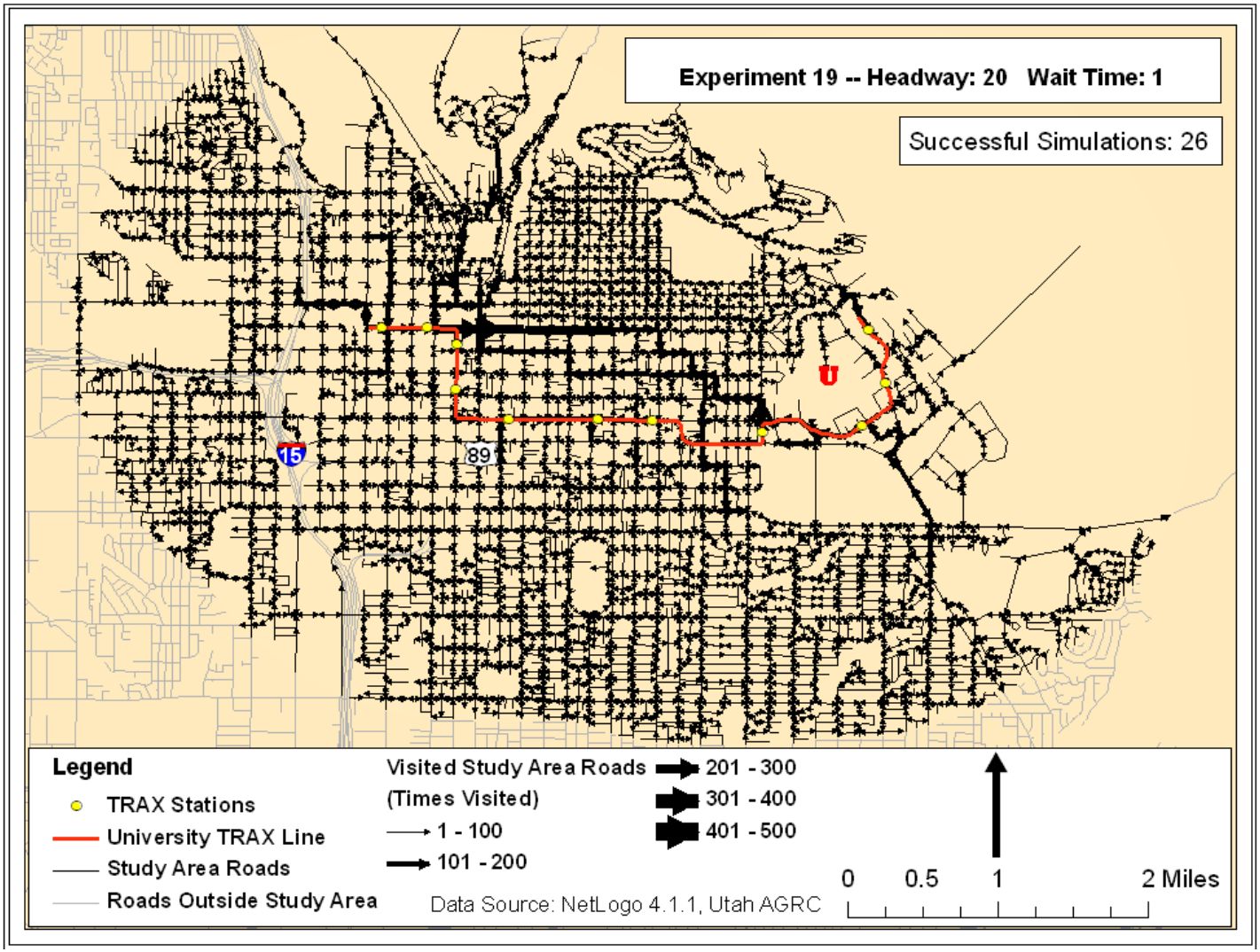

Figure 4.13 Map output of experiment 19: headway = 20 minutes and wait time = 1.

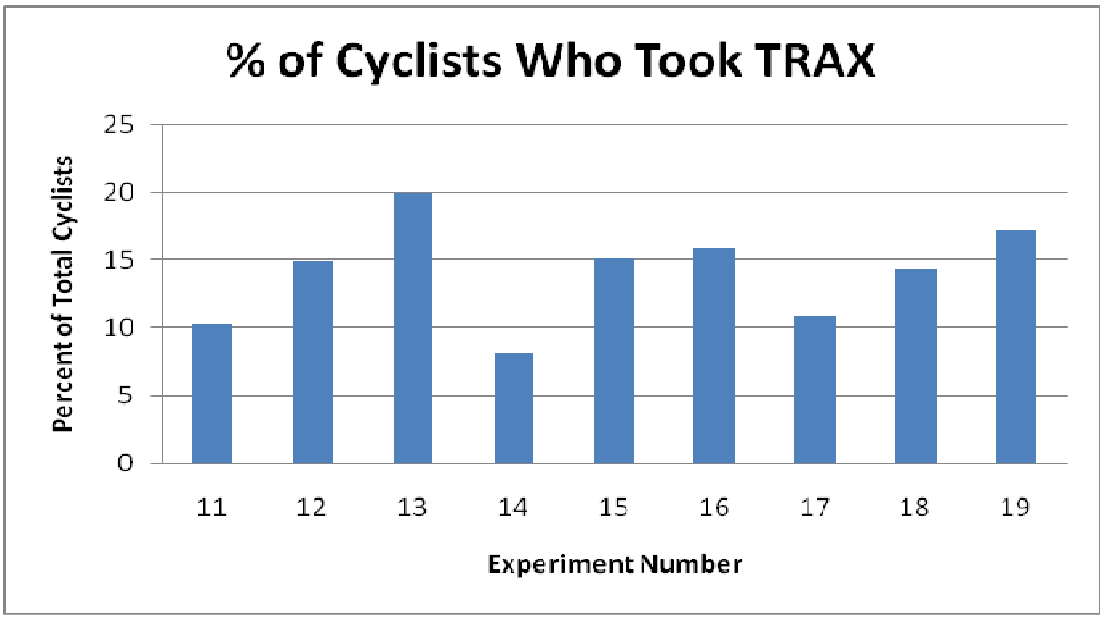

Figure 4.14 Histogram of the percentage of cyclists who took TRAX in set three.

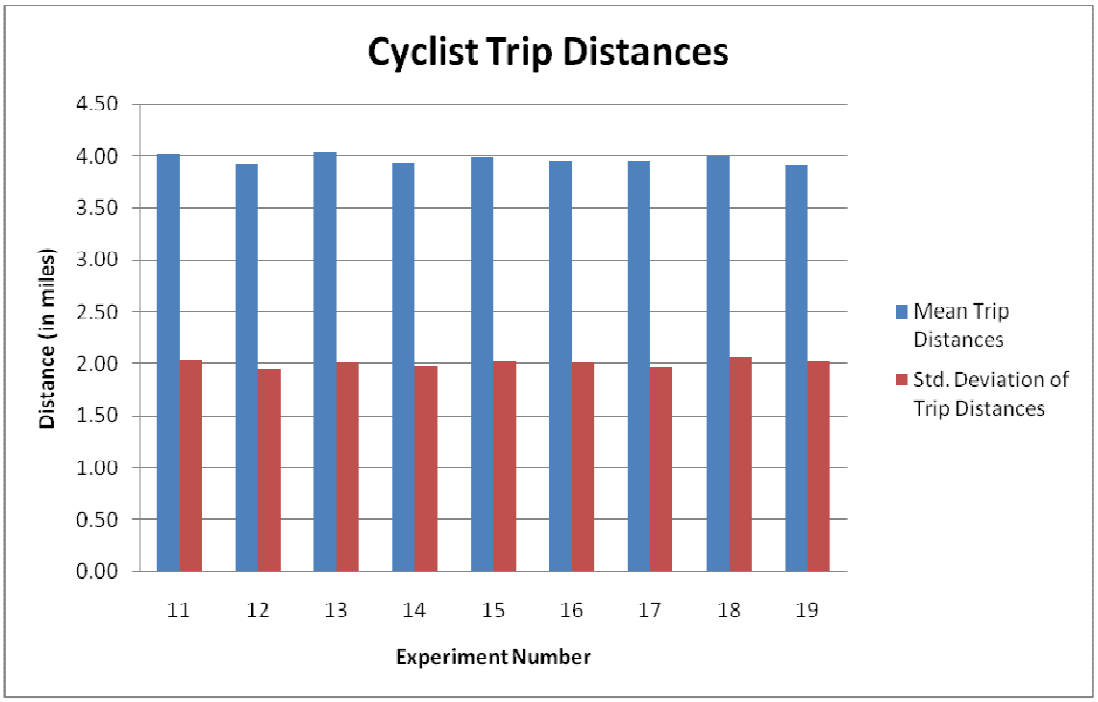

Figure 4.15 Histogram of the mean cyclist trip distances.

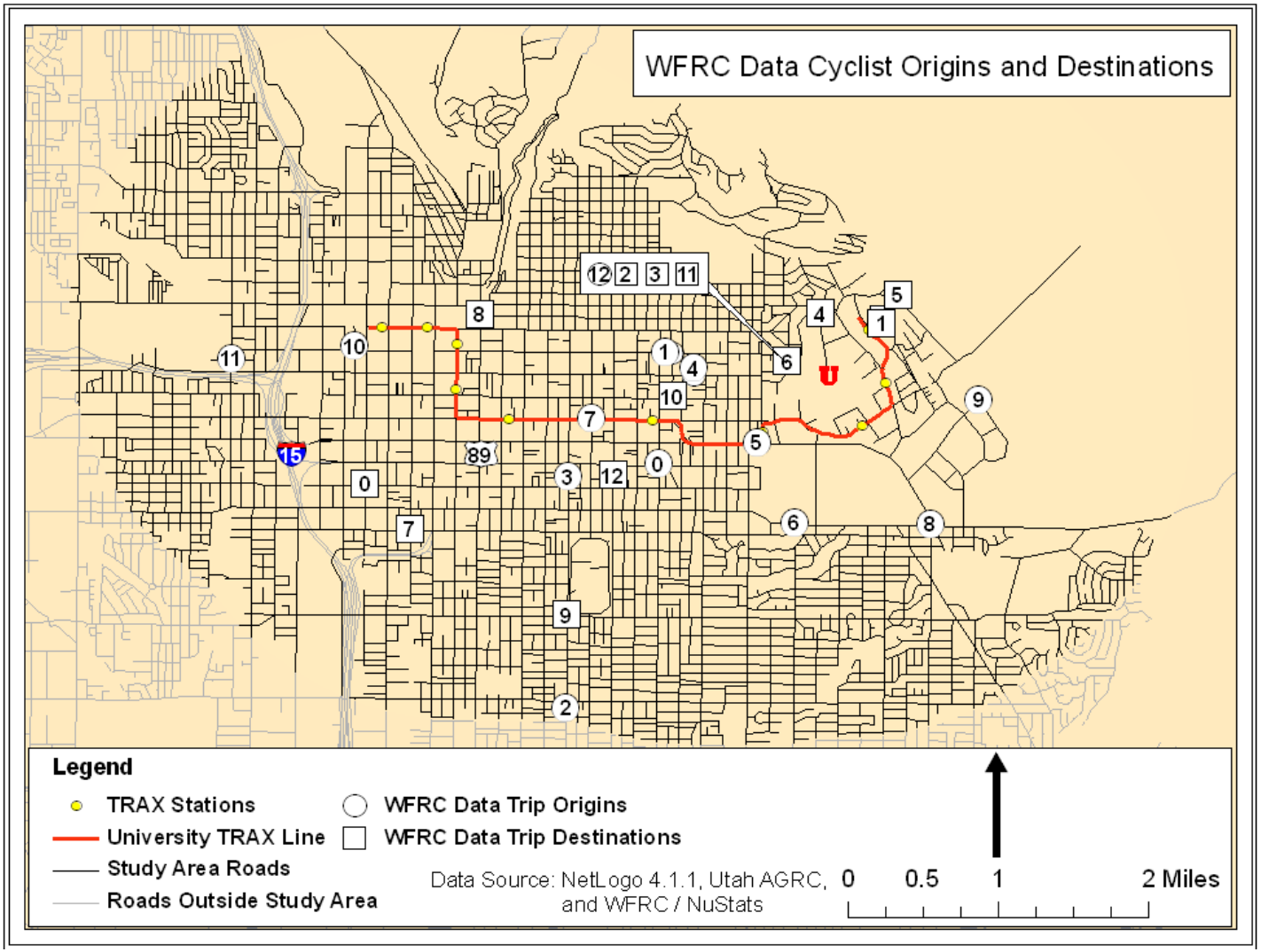

Figure 4.16 Map of the O/D locations of cyclists in the WFRC dataset.

Table 4.1 Number of successful simulations in the first set of experiments.

		Elevation (on / off)	
		on	*off*
Major Roads	*on*	34	27
	off	22	25

Table 4.2

Experiment Set One

Elevation versus Major Roads				
Experiment:	1	2	3	4
Simulation Parameters:	Elevation: Off Major Roads: Off	Elevation: Off Major Roads: On	Elevation: On Major Roads: Off	Elevation: On Major Roads: On
Number of Attempted Simulations:	38	35	38	35
Number of Successful simulations:	25	27	22	34
Number of Simulations with no output:	13	8	16	1
Total Cyclists:	2,500	2,700	2,200	3,400
Total Cycle-only trips:	2,168	2,376	1,829	2,869
Total Cycle + TRAX trips:	332	324	371	531
Mean cyclist trip distance: [a]	3.94 mi (20,792.47 ft)	3.95 mi (20,832.98 ft)	4.06 mi (21,413.34 ft)	4.01 mi (21,169.59 ft)
Mean standard deviation cyclist trip distance: [b]	2.02 mi (10,686.58 ft)	1.99 mi (10,484.35 ft)	2.06 mi (10,879.90 ft)	2.00 mi (10,562.63 ft)
Model Parameters				
Wait time:	0.8 (80%)			
Train headway:	15 minutes			
Cyclists following rules:	0.5 (50%)			

[a] The mean cyclist trip distance represents a grand mean of observed mean values per simulation run.

[b] The standard deviation cyclist trip distance represents a grand mean of observed mean standard deviation values per simulation run.

Table 4.3 Number of successful simulations in the second set of experiments.

		Cyclists Following Rules (yes / no)	
		no	*yes*
TRAX Headway	*10 minutes*	34	31
	15 minutes	33	35
	20 minutes	31	24

Table 4.4

Experiment Set Two

Train Headway versus Cyclists Following UTA Rules							
Experiment:		5	6	7	8	9	10
Simulation Parameters:	*Train Headway:*	10	10	15	15	20	20
	Following Rules:	No	Yes	No	Yes	No	Yes
Number of Attempted Simulations:		35	35	35	35	35	26
Number of Successful simulations:		34	31	33	35	31	24
Number of Simulations with no output:		1	4	2	0	4	2
Total Cyclists:		3,400	3,100	3,197	3,500	3,100	2,400
Total Cycle-only trips:		2,870	2,650	2,804	3,003	2,656	2,072
Total Cycle + TRAX trips:		530	450	393	497	444	328
Mean cyclist trip distance: [a]		3.95 mi (20,850.35 ft)	4.01 mi (21,197.93 ft)	4.06 mi (21,455.54 ft)	3.94 mi (20,799.11 ft)	3.91 mi (20,664.76 ft)	3.98 mi (20,998.11 ft)
Mean standard deviation cyclist trip distance: [b]		2.00 mi (10,552.89 ft)	1.99 mi (10,507.86 ft)	2.03 mi (10,731.52 ft)	2.02 mi (10,645.32 ft)	1.97 mi (10,422.92 ft)	2.01 mi (10,607.84 ft)
Model Parameters							
Wait time:		0.8 (80%)					
Elevation:		On					
Major Roads:		On					

[a] The mean cyclist trip distance represents a grand mean of observed mean values per simulation run.

[b] The standard deviation cyclist trip distance represents a grand mean of observed mean standard deviation values per simulation run.

Table 4.5 Number of successful simulations in the third set of experiments.

		Bicyclist Wait Time (% of TRAX Headway)		
		0.6	*0.8*	*1*
TRAX Headway	*10 minutes*	30	32	32
	15 minutes	24	22	27
	20 minutes	31	20	26

Table 4.6

Experiment Set Three

Train Headway versus Wait Time at TRAX Station

Experiment:	11	12	13	14	15	16	17	18	19
Simulation Parameters: *Train Headway:*	10	10	10	15	15	15	20	20	20
Wait Time(%):	0.6	0.8	1	0.6	0.8	1	0.6	0.8	1
Number of Attempted Simulations:	35	35	35	35	35	29	35	22	33
Number of Successful Simulations:	30	32	32	24	22	27	31	20	26
Number of Simulations with no output:	5	3	3	11	13	2	4	2	7
Total Cyclists:	3,000	3200	3200	2,357	2,178	2,664	3,100	2,000	2,600
Total Cycle-only trips:	2,691	2725	2562	2,165	1,849	2,242	2,762	1,712	2,153
Total Cycle + TRAX trips:	309	475	638	192	329	422	338	288	447
Mean cyclist trip distance: [a]	4.02 mi (21,248 ft)	3.93 mi (20,768 ft)	4.04 mi (21,313 ft)	3.94 mi (20,790 ft)	3.99 mi (21,091 ft)	3.95 mi (20,859 ft)	3.95 mi (20,876 ft)	4.00 mi (21,126 ft)	3.92 mi (20,698 ft)
Mean standard deviation cyclist trip distance: [b]	2.04 mi (10,772 ft)	1.96 mi (10,328 ft)	2.02 mi (10,687 ft)	1.98 mi (10,434 ft)	2.03 mi (10,724 ft)	2.01 mi (10,602 ft)	1.97 mi (10,415 ft)	2.07 mi (10,925 ft)	2.03 mi (10,705 ft)

Model Parameters

Cyclists following rules: 0.5 (50%) Elevation: : On Major Roads: On

[a] The mean cyclist trip distance represents a grand mean of observed mean values per simulation run.

[b] The standard deviation cyclist trip distance represents a grand mean of observed mean standard deviation values per simulation run.

5. CONCLUSION

The ABM used in this project utilizes a unique approach for modeling mode-choice behavior in a multimodal context. This ABM uses the NetLogo modeling environment, which shows that an ABM can be designed and implemented in a software package that is freely available. This section begins with an interpretation of the results presented in Sections 4.3 and 4.4. The second subsection briefly describes the limitations of the ABM. Subsection Three outlines the contributions of this project to the field of travel modeling, particularly in lieu of the stated objectives in Section 1. Subsection 5.4 concludes by offering some suggestions for future research directions.

5.1 Interpretation of Results

As noted in Sections 4.3 and 4.4, the output results of the simulations produced some results that were expected, while others were not as easily explained. One of the expected results was that assigning an arbitrarily high cost to some road segments, major roads in this case, caused the cyclists to find alternative routes. In experiments 1 and 3 where major roads were disallowed, the cyclists clearly avoid those road segments. In experiments 5 through 19, it was clear that the cyclists would use the major roads when they were allowed. This implies that the major road segments in this model are critical

links in many of the shortest paths throughout the network, particularly the section of Foothill Drive between 500 South and 1300 South.

Another expected result was from the third set of experiments, where cyclists were much more likely to use TRAX if they would wait for the entire duration of the train headway than if they were able to continue cycling before the train arrived. Figure 4.14 shows this clearly in the pattern of cyclists who used TRAX. Particularly when the train headway was 15 minutes and the cyclists would only wait for up to 60% of train headway, the percentage of cyclists using TRAX was quite low (just over 8%). On the other hand, if the cyclist waited for the entire time and the headway was only 10 minutes, TRAX ridership among cyclists jumped to 20%. This has a direct policy implication for UTA, in that it suggests fairly clearly that more frequent headways would lead to increased ridership among those who have alternative transportation options.

One of the unexpected outcomes of the simulations was the emergence of the 200 South to 700 East to 100 South corridors. While it was expected that a significant number of cyclists would be more willing to cycle downhill (east to west, in this case), this particular corridor did not seem to be the most likely for a cyclist to use. In reality, 200 South has a dedicated bicycle lane from State Street east to 1100 East, which makes it seem strange that a cyclist on 200 South would choose to head north to 100 South at 700 East. Of course, since bicycle lanes were not a component of the model, it is quite possible that this behavior would have been altered if bicycle lanes were included.

Another unexpected outcome of the results was the very small range in mean cyclist trip distances. Across all 19 experiments, the grand mean of cyclist trip distances was very close to 4.00 miles. Particularly under conditions more favorable to cyclists

riding TRAX, it would seem intuitive to have shorter cyclist trip distances. The cyclists were seeded at random locations and choose a random destination from a list of 50 possible destinations, so it is quite possible that trip distances would have been affected more profoundly if the model attempted to more realistically model real-world trips.

The most unexpected outcome of the experiments was the poor TRAX ridership among the cyclists in the tests of the WFRC dataset. While it was expected that not all of the cyclists represented by the WFRC would take TRAX every time, it is counterintuitive that only 1 of the 13 real-world cyclists took TRAX in the model under all simulation parameters. This implies that the model parameters are insufficient for capturing behavior that actually motivates a cyclist to use TRAX. Of course, the very small sample size of real-world cyclists is worth noting. If a comparison dataset had contained both cyclists who did not use transit as well as those who did, it would have been much easier to use data mining techniques to extract more realistic rules.

5.2 Limitations

While this ABM does show some promise for future usage (see Section 5.3), there are a few limitations worth noting. One of the biggest limitations is NetLogo itself. In the tests of the WFRC data when only 13 cyclists were moving around in the model, NetLogo did not return any error messages over the course of 180 simulation runs. When the model ran with 100 cyclists moving around, NetLogo returned several unexplained error messages. It seems likely that some of these errors were a result of a memory ceiling causing NetLogo to have insufficient power to process the complexities occurring

in the model. (This behavior in NetLogo was most noticeable in experiment 19, when train headways were 20 minutes and the cyclists waited at the station for the entire duration of the headway.) Additionally, since NetLogo has not been tested with very large models, particularly those using GIS data, it is also possible that there are some bugs in the GIS extension source code that cause the extremely long processing time.

Another limitation of these results is the comparatively small number of simulation runs. For users to extract very meaningful conclusions for a model such as this, it would be more useful to run several hundred simulations to obtain a clearer picture of the breadth of factors influencing the behavior of cyclists moving throughout the model. Since the model typically ran one simulation in 6 to 8 minutes, the time to process the data would be unacceptably long. Assuming 999 simulations for each of the 19 experiments with a run time of 8 minutes per simulation, it would take 106 days to complete all simulation runs with no errors and no stoppages in running the simulations. The large amount of data that this many simulations would generate would then take a very long time to process as well. Since the purpose of this project was based on the proof-of-concept that an ABM is suitable for solving this sort of problem, little additional effort went in to optimizing the model code to get it to run as quickly as possible. Thus, some time-based gains would be seen if the code were adjusted to run as efficiently as possible.

Lastly, the study area is fairly small compared to the overall area where people frequently cycle in the Salt Lake Valley. It is likely that there are many cyclists who cycle into the study area for work during the day and then cycle back to a home destination outside of the study area. Additionally, since the existing TRAX network

extends south as far Sandy and will soon extend out to West Valley City, West Jordan, Draper and the Salt Lake International Airport, a model such as this that focuses on multimodal behavior should use a slightly larger study area that captures more of the area where multiple modes are accessible.

5.3 Contributions of this Model

In Section 1, three primary objectives for this project were: (1) showing the usefulness of agent-based modeling to solve a mode-choice problem, (2) constructing an ABM using available software, and (3) to show that the results obtained from an ABM are useful such that using an ABM offers perspectives on mode-choice that are unavailable through traditional modeling techniques. Regarding the first objective, it seems fairly clear that an ABM is a viable method for capturing multimodal behavior. As shown in Section 4.3, the model parameters were effective in affecting the percentage of cyclists who took TRAX as part of their trip. The various experiments produced a range of 8% to 20% of cyclists who made multimodal trips. Thus, it seems evident that using ABM can help meet a stated goal of the TRB report, namely, focusing on multimodal trips in travel models (TRB Special Report 288, 2007).

In terms of the second objective, NetLogo has proven to be a possible solution for a viable modeling environment, though it seems evident that NetLogo may not be the *best* environment for tackling this sort of problem. While model optimization could achieve better processing time, it would seem that a more appropriate solution would be to use software more tightly-coupled with a GIS. This would allow the enormity of data

to be handled by a software package that is well-suited to handling large, spatial datasets. Other ABM platforms exist, such as RePast, and these packages could also be viable solutions. One of the primary benefits of using NetLogo is that it is more accessible to those with fewer programming skills, which implies that other environments would have a steeper learning curve. Even if a software package is freely available, a steep learning curve in terms of time spent understanding how to produce a model could prove cost-prohibitive for many planning agencies with limited resources.

In terms of the third objective, this ABM clearly provides useful results that are unavailable from traditional modeling methods. It is likely that the unexpected 200 South to 700 East to 100 South corridors appeared as a result of elevation differences. The first set of experiments also showed that assigning a high cost to some roads caused the cyclists to avoid those routes. One of the more conclusive pieces of evidence for useful ABM outputs is Figure 4.14, which shows that the parameters of train headway and bicyclist wait time had a very distinct impact on cyclists choosing TRAX. While traditional methods could potentially model headway and wait time parameters, an ABM captures these behaviors at an individual level that most other modeling methods cannot achieve. Thus, this ABM has proven quite useful in solving a mode-choice problem and producing results that traditional methods would have difficulty replicating. The NetLogo modeling environment may not be the most ideal for this sort of model (with thousands of nodes), but it does show that a model such as this can be constructed and used in a software package that is freely available.

5.4 Future Areas of Investigation

While this project has shown the usefulness and viability of ABM, there is much work left to be done. As noted earlier, it would be useful to try constructing an ABM in an environment that is tightly-coupled with a GIS. At the release of ArcGIS version 10, the Python scripting language was more heavily integrated into GIS operations. It is quite possible that the use of Python in ArcGIS could produce a model that could make use of the existing network routing applications within the ArcGIS Network Analyst and allow for the movement of mobile objects in more of a simulation format than the traditional, static map produced in ArcGIS. It seems likely that the use of a modeling platform more tightly-coupled with a GIS could produce much quicker simulation run times.

Particularly in the Salt Lake City area, it would be useful to build a model that includes other multimodal possibilities, such as bike-on-bus and cycling to a TRAX station and renting one of the many available bicycle lockers. As is the case with most models, the more input data that are included in the model, the more refined (and accurate) the results would be. Of course, more model complexity would lead directly to higher processing costs and more computational power. If the model can be optimized so that the processing time is drastically reduced, then more model parameters and options become viable.

It is well-known in the transportation planning community that the transportation infrastructure is closely tied with land-use in terms of where trips are generated and distributed. Land-use is the major component missing from this model, and so it is necessary to build in a land-use component to this model to gain more acceptance in the

planning and engineering communities. In this existing model, one would need to determine a method for creating a land-use component for each street network node. This sort of data would be derived from sources such as the National Land Cover Dataset and County Assessor's data. Since a large number of land-use / transportation models already exist, it would be most useful to tie in a multimodal ABM with one of these existing models to reduce model production costs. Thus, this model is a very good launching point for more work to be done. As the research grows in using ABM for transportation-based applications, the models will only get better and become more widespread in the transportation community.

Finally, it would be best to test this model using a more appropriate dataset. The ideal dataset for this model would be a comprehensive survey of bicyclists in the Salt Lake City area. This would need to include both TRAX users and those who do not use any other modes. Some of the data collected would need to be similar to that of the WFRC data. A comprehensive survey of cyclists could be data-mined to determine appropriate rules for the model. This would allow the model to more accurately represent cyclist behavior and cause the outputs of the model to predict future cyclist behavior with reasonable accuracy. To obtain this type of dataset, it would be reasonable to partner with the Salt Lake Bicycle Collective and use its network of local cyclists to promote cyclist involvement with the survey. The survey itself would need to be conducted at a stationary location (such as the Bike Collective) rather than onboard transit vehicles (as was the case with the WFRC data). The survey could also be administered online, which may promote higher cyclist response rates. A more appropriate dataset would allow the

model to move from being a novel concept to a useful planning tool when taken in context.

APPENDIX

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;; AN AGENT-BASED MODEL OF BICYCLISTS ACCESSING ;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;LIGHT RAIL IN SALT CITY,UTAH;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;; Author: Josh Groeneveld, University of Utah Department of Geography
;;;;;; Date:   Spring 2011
;;;;;; NetLogo Version: 4.1.1

;;;;;; Special thanks to David M. Johnson of the University of Utah School of Computing
;;;;;; Flux Research Group.  David's generous and patient help made this model possible.
;;;;;; Thanks, David!

;;;;;; Special thanks also to William John Teahan.  The path-finding algorithm used in this
;;;;;; model is derived from his version of the Dijkstra algorithm in the Being Kevin
;;;;;; Bacon model.
;;;;;; Being Kevin Bacon NetLogo model.
;;;;;; Teahan, W. J. (2010). Artificial Intelligence. Ventus Publishing Aps.
;;;;;; Copyright 2010 by William John Teahan.  All rights reserved.

;;;;;; These functions declare the extensions used, agent breeds (links and turtles), and
;;;;;; agent-specific variables.
extensions [ gis array ]

breed [nodes node]
breed [people person]
breed [stations station]
breed [trains train]

directed-link-breed [trax_lines trax_line]
directed-link-breed [streets street]
links-own [
  cost ; Cost represents the arc weight for all links (TRAX lines and streets).
  time ; Time represents the number of seconds a cyclist or train would take to traverse a
link.
  dist ; Dist represents the distance in feet from one end of the link to the other.
```

```
  speed ; Speed represents the speed an agent can traverse the link (in feet / second).
  beta ; Beta represents the conversion factor to go from NetLogo space distances to real-
world distances.
  slope ; Slope represents the slope of the link.
  ]
nodes-own [
  myelev ; The elevation of the patch the node is centered on, based on the LiDAR data.
  closest-station ; The closest TRAX station to the node, based on a GIS layer of Theissen
polygons around stations.
  is-station ; Indicates if the node is topologically coincident with a TRAX station.
  seed-time ; A randomly generated number indicated when a cyclist spawned at the node
may start moving.
  id ; A consistent identifier derived from the GIS data.
  ]
people-own [
  location ; The cyclist's origin node.
  new-location ; The cyclist's destination node.
  current-location ; The cyclist's current node.
  dijkstra-distances ; An array containing distances based on the Dijkstra algorithm to all
nodes in the network.
  dijkstra-directions ; An array containing directions to all reachable nodes in the network
based on Dijkstra.
  dijkstra-from-trax-directions ; An array representing directions to all reachable nodes
from the destination TRAX.
  dijkstra-from-o-trax-directions ; An array representing directions to all reachable nodes
from the origin TRAX.
  dijkstra-from-o-trax-distances ; An array representing distances to all reachable nodes
from the origin TRAX.
  nodes-visited ; A list of nodes reached during the application of the Dijkstra algorithm.
  still-to-visit ; A list of nodes the cyclist needs to visit in order to reach its destination.
  bike-ticks-to-wait ; The number of ticks (seconds) a cyclist needs to wait before it
moves to the next node.
  bike-time-to-move ; The number the tick counter must reach for the cyclist to move.
  closest-o-trax-station ; The closest TRAX station to the cyclist's origin.
  closest-d-trax-station ; The closest TRAX station to the cyclist's destination.
  bike-or-trax-path ; A string representing whether the cyclist is pursuing a cycle-only
path or a TRAX-based path.
  bike-direction ; The cyclist's direction (east or west).
  trax-path-cost ; The cost for the cyclist to continue to destination via a TRAX-based
path.
  rule-follower ; A binary variable indicating whether the cyclist is observing UTA rules.
  my-train ; The train the cyclist is waiting to board at the TRAX station.
  riding-train? ; A boolean variable representing whether or not the cyclist is actually on
the train.
```

```
  nodes-visited-list ; A list of nodes the cyclist visits--used for path re-creation in post-processing.
  time-visited-list ; A list of the time steps when the cyclists visited nodes--used in post-processing.
  dist-list ; A list containing the distance of the links the cyclist has crossed.
  trax-ride-cost ; A number representing just the cost for the TRAX portion of the trip.
  trax-dist ; The distance (in feet) a cyclist would need to travel on TRAX.
  bike-type ; This represents whether a cyclist initially pursued a cycle-based path, or if it made this decision at the origin TRAX station.
  ]
trains-own [
  current-stop ; The train's current station.
  bike-cap ; The train's available capacity for cyclists.
  train-num ; The train's UTA number (a unique identifier).
  direction ; The train's direction (east or west).
  next-stop ; The train's next station.
  stops-to-visit ; A list of stations the train needs to visit before reaching the end of the line.
  ticks-to-wait ; The number of ticks (seconds) the train must wait before it moves.
  time-to-move ; The time the tick counter needs to reach before the train moves.
  initial-time ; The initial time a train must wait before moving (used only at the beginning of each simulation).
  stops-visited ; The number of stations a train has visited (useful for debugging).
  initial-location ; The train's initial (seed) location.
  ]
streets-own [
  major-road ; Indicates whether or not a street is a major road (based on GIS data).
  elevation-time ; The time a cyclist would take to traverse the street with the elevation considered.
  non-elev-time ; The time a cyclist would take to traverse the street with the elevation (slope) not considered.
  ]
stations-own [
  eb-sta-num ; The station's number from west to east.
  wb-sta-num ; The stations' number from east to west.
  wait-time ; The amount of time a train must wait at this station (60 seconds for stations, 0 for pseudo-stations).
  eb-time ; The time to the next station headed eastbound.
  wb-time ; The time to the next station headed westbound.
  eb-tcounter ; The time (in seconds) til the next eastbound train reaches this station.
  wb-tcounter ; The time (in seconds) til the next westbound train reaches this station.
  next-eb-train ; The next train to reach this station headed eastbound.
  next-wb-train ; The next train to reach this station headed westbound.
  initial-eb-tcounter ; The initial time (in seconds) to the next eastbound train.
  initial-wb-tcounter ; The initial time (in seconds) to the next westbound train.
```

```
  station-name ; The name of this station, based on UTA station names.
  counter-list
  eb-sta-time
  wb-sta-time
  dijkstra-station-distances ; An array representing distances to all nodes from this station.
  dijkstra-station-directions ; An array representing directions to all nodes from the
station.
  station-nodes-visited ; The nodes visited during the Dijkstra procedure.
]
globals [
  infinity ; A large number.
  intersections ; Used for importing the node GIS data.
  roadset ; Used for importing the street GIS data.
  roads ; Also used for importing the street GIS data.
  err_nobody ; An error that occurs when a link cannot be created because one of the end
nodes does not exist.
  err_same_turtle ; An error that occurs when there are multiple nodes on the same patch.
  num_roads ; The number of streets created when the model initializes.
  trax_lineset ; Used for importing the TRAX line GIS data.
  count_stations ; The number of stations and pseudo-stations in the model.
  junctions ; Used for importing the TRAX staiton GIS data.
  traxlines ; Also used for importing the TRAX line GIS data.
  o-nodes ; A list of possible origin nodes (100 nodes for each simulation.)
  d-nodes ; A list of possible destination nodes (50 nodes for each simulation.)
  elevation ; Used for importing the LiDAR elevation data.
  closest-sta-list ; Used for importing the closest TRAX station GIS data variable.
  is-sta-list ; Used for importing the GIS data variable indicting whether a node is
coincident with a TRAX station.
  current_envelope ; The current envelope of the NetLogo world.
  numlist ; Used for assigning the TRAX train identifier number.
  eb-sta-order ; Used for importing the GIS data variable for eastbound TRAX station
order.
  wb-sta-order ; Used for importing the GIS data variable for westbound TRAX station
order.
  eb-sta-list ; Also used for importing the GIS data variable for eastbound TRAX station
order.
  wb-sta-list ; Also used for importing the GIS data variable for westbound TRAX station
order.
  wait-list ; Used for importing the GIS data variable for TRAX station wait time.
  next-east-list ; Used for importing the GIS data variable for the next eastbound TRAX
station.
  next-west-list ; Used for importing the GIS data variable for the next westbound TRAX
station.
  name-list ; Used for importing the GIS data variable for the TRAX station name.
```

```
  eb-trax-cost ; An array representing the TRAX path cost from one station to any other
station in the eastbound direction.
  wb-trax-cost ; An array representing the TRAX path cost from one station to any other
station in the westbound direction.
  eb-trax-dist ; An array representing the TRAX path distance from one station to any
other headed eastbound.
  wb-trax-dist ; An array representing the TRAX path distance from one station to any
other headed westbound.
  num-bicyclists ; A value used in post-processing representing the total number of cycle-
only trips per simulation.
  num-trax-riders ; A value used in post-processing representing teh total number of
TRAX-based trips per simulation.
  sum-cyclist-cost ; The sum of the cost for all links traversed by cyclists.
  sum-cyclist-dist ; The sume of the distance traveled for all links traversed by cyclists.
  id-list ; Used in post-processing...converts node IDs into useable numbers for mapping
output in GIS.
  sim-num ] ; Represents the simulation number, used in post-processing.

to startup ; A procedure called only when NetLogo first opens.  These commands run
only once.
  initialize-model
end

to setup ; Prepares each simulation run by resetting the model, spawning cyclists and
computing paths.
  reset-model
  prepare-cyclists
end

to initialize-model ; Opens the model, loads in GIS data and translates it into NetLogo
space.
  ca
  set infinity 65535
  gis:load-coordinate-system "SLC_2mi_Road_Junctions_update.prj"
  show "coords loaded"
  set intersections gis:load-dataset "SLC_2mi_Road_Junctions_update.shp"
  show "intersections loaded"
  set roadset gis:load-dataset "SLC_Roads_2mi_UTrax_update.shp"
  show "roadset loaded"
  set elevation gis:load-dataset "lidar_125_large.asc"

  set junctions gis:load-dataset "University_Trax_station_nodes.shp"
  show "junctions loaded"
  set trax_lineset gis:load-dataset "Trax_line2.shp"
  show "trax_lineset loaded"
```

```
resize-world 0 gis:width-of elevation 0 gis:height-of elevation
gis:set-world-envelope (gis:envelope-union-of (gis:envelope-of intersections)
                                              (gis:envelope-of roadset)
                                              (gis:envelope-of elevation))

ask patches [ set pcolor black ]
ask patches gis:intersecting intersections
[ set pcolor cyan ]
ask patches with [ pcolor = cyan]
[sprout-nodes 1]

let node-stations gis:feature-list-of intersections
foreach node-stations [
  let cs gis:property-value ? "STATIONNAM"
  let is gis:property-value ? "STATION"
  let fid gis:property-value ? "FID_SLC_2M"

  foreach (gis:vertex-lists-of ?) [
  let v1 (first ?)
  let x1 (first (gis:location-of v1))
  let y1 (last (gis:location-of v1))

  let t1 one-of ((nodes-on patch x1 y1))
  set closest-sta-list []
  ask t1 [set closest-sta-list fput cs closest-sta-list]
  ask t1 [set closest-station item 0 closest-sta-list]
  set is-sta-list []
  ask t1 [set is-sta-list fput is is-sta-list]
  ask t1 [set is-station item 0 is-sta-list]
  set id-list []
  ask t1 [set id-list fput fid id-list]
  ask t1 [set id item 0 id-list]
  ]
]

set err_nobody 0
set err_same_turtle 0
set num_roads 0

set roads gis:feature-list-of roadset
foreach roads [
  let oneway gis:property-value ? "ONE_WAY"
  let major gis:property-value ? "ALT_NAME"
```

```
foreach (gis:vertex-lists-of ?) [
let v1 (first ?)
let v2 (last ?)
let x1 (first (gis:location-of v1))
let y1 (last (gis:location-of v1))
let x2 (first (gis:location-of v2))
let y2 (last (gis:location-of v2))

let t1 one-of ((nodes-on patch x1 y1))
let t2 one-of ((nodes-on patch x2 y2))

if-else (t1 != t2 and t1 != NOBODY and t2 != NOBODY) [
  let t3 [who] of t1
  let t4 [who] of t2
  if (oneway = 1 and major = "") [
    ask t1 [create-street-to t2
      ask street t3 t4 [
        set major-road 0]
    ]
  ]
  if (oneway = 2 and major = "") [
    ask t2 [create-street-to t1
      ask street t4 t3 [
        set major-road 0]
    ]
  ]
  if (oneway = 0 and major = "") [
    ask t1 [create-street-to t2
      ask street t3 t4 [
        set major-road 0]
    ]
  ]
  if (oneway = 0 and major = "") [
    ask t1 [create-street-from t2
      ask street t3 t4 [
        set major-road 0]
    ]
  ]
  if (oneway = 1 and major != "") [
    ask t1 [create-street-to t2
      ask street t3 t4 [
        set major-road 1]
    ]
  ]
  if (oneway = 2 and major != "") [
```

```
      ask t2 [create-street-to t1
        ask street t4 t3 [
          set major-road 1]
      ]
    ]
    if (oneway = 0 and major != "") [
      ask t1 [create-street-to t2
        ask street t3 t4 [
          set major-road 1]
      ]
    ]
    if (oneway = 0 and major != "") [
      ask t1 [create-street-from t2
        ask street t4 t3 [
          set major-road 1]
      ]
    ]
    set num_roads (num_roads + 1)
  ] [
    if (t1 = t2) [
      set err_same_turtle (err_same_turtle + 1)
    ]

    if-else (t1 != NOBODY) [
    ] [
      set err_nobody (err_nobody + 1)
    ]

    if-else (t2 != NOBODY) [
    ] [
      set err_nobody (err_nobody + 1)
    ]
  ]
  ]
]

ask patches gis:intersecting junctions
[ set pcolor orange ]
ask patches with [ pcolor = orange]
[sprout-stations 1]

let station-nodes gis:feature-list-of junctions
foreach station-nodes [
  let name gis:property-value ? "STATIONNAM"
  let wb gis:property-value ? "WB"
```

```
  let eb gis:property-value ? "EB"
  let wt gis:property-value ? "WAIT"
  let ne gis:property-value ? "NEXTEAST"
  let nw gis:property-value ? "NEXTWEST"

  foreach (gis:vertex-lists-of ?) [
  let v1 (first ?)
  let x1 (first (gis:location-of v1))
  let y1 (last (gis:location-of v1))

  let t1 one-of ((stations-on patch x1 y1))
  set eb-sta-list []
  ask t1 [set eb-sta-list fput eb eb-sta-list]
  ask t1 [set eb-sta-num item 0 eb-sta-list]
  set wb-sta-list []
  ask t1 [set wb-sta-list fput wb wb-sta-list]
  ask t1 [set wb-sta-num item 0 wb-sta-list]
  set wait-list []
  ask t1 [set wait-list fput wt wait-list]
  ask t1 [set wait-time item 0 wait-list]
  set next-east-list []
  ask t1 [set next-east-list fput ne next-east-list]
  ask t1 [set eb-tcounter item 0 next-east-list]
  set next-west-list []
  ask t1 [set next-west-list fput nw next-west-list]
  ask t1 [set wb-tcounter item 0 next-west-list]
  set name-list[]
  ask t1 [set name-list fput name name-list]
  ask t1 [set station-name item 0 name-list]
  ]
]

ask stations [
  if eb-tcounter = NOBODY [
    set eb-tcounter 0
  ]
  if wb-tcounter = NOBODY [
    set wb-tcounter 0
  ]
]

set traxlines gis:feature-list-of trax_lineset
foreach traxlines [
  foreach (gis:vertex-lists-of ?) [
  let v1 (first ?)
```

```
  let v2 (last ?)
  let x1 (first (gis:location-of v1))
  let y1 (last (gis:location-of v1))
  let x2 (first (gis:location-of v2))
  let y2 (last (gis:location-of v2))

  let t1 one-of ((stations-on patch x1 y1))
  let t2 one-of ((stations-on patch x2 y2))

  if-else (t1 != t2 and t1 != NOBODY and t2 != NOBODY) [
    ask t1 [create-trax_line-to t2]
    ask t1 [create-trax_line-from t2]
    set num_roads (num_roads + 1)
  ] [
    if (t1 = t2) [
      set err_same_turtle (err_same_turtle + 1)
    ]

    if-else (t1 != NOBODY) [
    ] [
      set err_nobody (err_nobody + 1)
    ]
    if-else (t2 != NOBODY) [
    ] [
      set err_nobody (err_nobody + 1)
    ]
  ]
  ]
]

ask nodes [set shape "dot"]
ask nodes [set color blue
  set size 3]
ask streets [set color red]
ask streets [show-link]
ask nodes [ set pcolor green]
gis:paint elevation 0
ask nodes [set myelev gis:raster-value elevation xcor abs (625 - ycor)]
ask streets [
 set beta 52.73
 set dist (link-length * beta)
 set slope ((([myelev] of end2 - [myelev] of end1) / dist) * 100)
 if slope > 5 [set speed 440]
 if slope > 2 and slope <= 5 [set speed 880]
 if slope > -2 and slope <= 2 [set speed 1056]
```

```
 if slope > -5 and slope <= -2 [set speed 1320]
 if slope <= -5 [set speed 1760]
 set time (dist / speed)
 set time (time * 60)
 set time round time
 set elevation-time time
 let non-elev-speed 1056
 set non-elev-time (dist / non-elev-speed)
 set non-elev-time (non-elev-time * 60)
 set non-elev-time round non-elev-time
]
ask trax_lines [
  set beta 52.73
  set dist (link-length * beta)
  set speed 1836
  set time (dist / speed)
  set time (time * 60)
  set time round time
]

ask stations [set shape "dot"]
ask stations [set color yellow]

ask stations [
  set counter-list []
  if station-name = "Arena" [
    set counter-list [ 600 360 720 600 600 960 ] ]
  if station-name = "Temple Square" [
    set counter-list [ 480 480 600 720 480 1080 ] ]
  if station-name = "City Center" [
    set counter-list [ 360 600 480 900 360 1200 ] ]
  if station-name = "Gallivan Plaza" [
    set counter-list [ 240 120 360 120 240 120 ] ]
  if station-name = "Library" [
    set counter-list [ 600 300 120 300 1200 300 ] ]
  if station-name = "Trolley" [
    set counter-list [ 480 420 900 420 1080 420 ] ]
  if station-name = "900 East" [
    set counter-list [ 360 540 780 540 960 540 ] ]
  if station-name = "Stadium" [
    set counter-list [ 180 120 600 720 780 720 ] ]
  if station-name = "University South Campus" [
    set counter-list [ 60 300 480 900 660 900 ] ]
  if station-name = "Fort Douglas" [
    set counter-list [ 540 420 360 120 540 1020 ] ]
```

```
    if station-name = "University Medical Center" [
      set counter-list [ 420 600 240 300 420 1200 ] ]
    if station-name = "" [
      set counter-list [ 0 0 0 0 0 0 ] ]
  ]

  ask stations [
    set initial-eb-tcounter eb-tcounter
    set initial-wb-tcounter wb-tcounter
  ]
  ask stations [
    set wb-sta-time array:from-list n-values (count stations + 1) [count stations]
    set eb-sta-time array:from-list n-values (count stations + 1) [count stations]
  ]
  ask stations [
    set-time-array
  ]

  set sim-num 0

  type "err_same_turtle count: " type err_same_turtle print ""
  type "err_nobody count: " type err_nobody print ""
  type "num_roads count: " type num_roads print ""
end

to set-time-array ; Sets up the time arrays to travel between train stations.
  array:set wb-sta-time [wb-sta-num] of self 0
  array:set eb-sta-time [eb-sta-num] of self 0
  foreach ([who] of stations) [
    ask station ? [
      let eb [eb-sta-num] of self
      let wb [wb-sta-num] of self
      let i 1
      let j 1
      while [ i < eb ] [
        array:set eb-sta-time i 9999
        set i (i + 1)
      ]
      while [ j < wb ] [
        array:set wb-sta-time j 9999
        set j (j + 1)
      ]
      let time-to-next 0
      while [ eb < 31 ] [
        let a item 0 [who] of stations with [eb-sta-num = eb]
```

```
      let b item 0 [who] of stations with [eb-sta-num = eb + 1]
      set time-to-next [time] of trax_line a b
      array:set eb-sta-time (eb + 1) time-to-next
      set eb eb + 1
     ]
     while [ wb < 31 ] [
      let c item 0 [who] of stations with [wb-sta-num = wb]
      let d item 0 [who] of stations with [wb-sta-num = wb + 1]
      set time-to-next [time] of trax_line c d
      array:set wb-sta-time (wb + 1) time-to-next
      set wb wb + 1
     ]
   ]
  ]
end

to initialize-stops-to-visit ; Used to initialize the stops that trains need to visit to complete
their trips.
   set stops-to-visit []
   set stops-to-visit fput current-stop stops-to-visit
   let ns 0
   set ns current-stop
   while [ns < 32] [
    set stops-to-visit fput ns stops-to-visit
    set ns (ns + 1)
   ]
   set stops-to-visit reverse stops-to-visit
   set stops-to-visit remove-duplicates stops-to-visit
   set stops-to-visit remove-item 0 stops-to-visit

  set wb-sta-order array:from-list n-values (count stations + 1) [count stations]
  set eb-sta-order array:from-list n-values (count stations + 1) [count stations]
  ask stations [
   array:set wb-sta-order [wb-sta-num] of self [who] of self
   array:set eb-sta-order [eb-sta-num] of self [who] of self
  ]
end

to calculate-time-to-move ; Called by trains to determine what time they can move to the
next stop.
   set ticks-to-wait 0
   let cs item 0 [who] of stations-here
   if-else direction = "east" [
    let nes array:item eb-sta-order (item 0 stops-to-visit)
    set ticks-to-wait ([time] of trax_line cs nes + item 0 [wait-time] of stations-here)]
```

```
    [ let nws array:item wb-sta-order (item 0 stops-to-visit)
      set ticks-to-wait ([time] of trax_line cs nws + item 0 [wait-time] of stations-here)]
    set time-to-move (ticks + ticks-to-wait)
end

to change-directions ; Called by trains to begin a new trip after reaching the end of the
line.
      if-else direction = "east" [
      set direction "west"
      set current-stop item 0 [wb-sta-num] of stations-here]
    [ set direction "east"
      set current-stop item 0 [eb-sta-num] of stations-here]
    initialize-stops-to-visit
end

to reset-model ; Clears output from the previous simulation and ensures a clean setup for
the following simulation.
  ask people [ die ]
  ask trains [ die ]
  reset-ticks
  clear-output
  ask nodes [ set color blue ]
end

to prepare-cyclists ; Spawns cyclists and calculates shortest paths to nearest TRAX
station and directly to destination.
  ask nodes [
    set seed-time random 3600 ; This ensures that all cyclists begin moving within one
simulated hour of the start of the simulation.
  ]
  let non-station-nodes nodes with [is-station = "No"]
  ask n-of 100 non-station-nodes [hatch-people 1]
  set o-nodes []
  ask people [set o-nodes fput nodes-here o-nodes]
;  print o-nodes
  set d-nodes []
  set d-nodes fput n-of 50 nodes d-nodes
;  print d-nodes
  ask people [set location one-of nodes-here]
  ask people [set new-location one-of d-nodes]
  ask people [set new-location one-of new-location]
  ask people [set shape "bike"
    set size 3]
  ask streets [
    if-else elevation-on? = "yes" [
```

```
    set cost (((1 - alpha) * link-length) + (alpha * elevation-time))
  ]
  [ set cost (((1 - alpha) * link-length) + (alpha * non-elev-time))
  ]
]
ask streets with [major-road = 1] [
  if-else major-roads-on? = "yes" [
    set cost (cost * 1)
    if hidden? = True [
      show-link]
  ]
  [ set cost 1999
    hide-link]
]
ask trax_lines [ set cost (((1 - alpha) * link-length) + (alpha * time))
]

ask stations [
  if train-headway = 15 [
    set numlist [ 41 45 42 44 43 ]
    if station-name = "Arena" [
      hatch-trains 1
    ]
    if station-name = "Trolley" [
      hatch-trains 1
    ]
    if station-name = "University Medical Center" [
      hatch-trains 1
    ]
    if station-name = "University South Campus" [
      hatch-trains 1
    ]
    if station-name = "City Center" [
      hatch-trains 1
    ]
    let n 0
    foreach sort-by [[xcor] of ?1 < [xcor] of ?2] trains [
      ask ? [ set train-num item n numlist
        set n n + 1]
    ]
    ask trains [
      if-else train-num = 42 or train-num = 43 [
        set direction "west"]
      [ set direction "east"]
    ]
```

```
]
if train-headway = 10 [
  set numlist [ 41 46 42 45 43 44 ]
  if station-name = "Arena" [
    hatch-trains 1
  ]
  if station-name = "Library" [
    hatch-trains 1
  ]
  if station-name = "University South Campus" [
    hatch-trains 1
  ]
  if station-name = "University Medical Center" [
    hatch-trains 1
  ]
  if station-name = "Stadium" [
    hatch-trains 1
  ]
  if station-name = "City Center" [
    hatch-trains 1
  ]
  let n 0
  foreach sort-by [[xcor] of ?1 < [xcor] of ?2] trains [
    ask ? [ set train-num item n numlist
      set n n + 1]
  ]
  ask trains [
    if-else train-num = 42 or train-num = 43 or train-num = 44 [
      set direction "west"]
    [ set direction "east"]
  ]
]
if train-headway = 20 [
  set numlist [ 43 41 42 ]
  if station-name = "City Center" [
    hatch-trains 1
  ]
  if station-name = "Library" [
    hatch-trains 1
  ]
  if station-name = "University Medical Center" [
    hatch-trains 1
  ]
  let n 0
  foreach sort-by [[xcor] of ?1 < [xcor] of ?2] trains [
```

```
      ask ? [ set train-num item n numlist
        set n n + 1]
    ]
    ask trains [
      if-else train-num = 41 or train-num = 42 [
        set direction "west"]
      [ set direction "east"]
    ]
  ]
]

ask trains [
  set shape "train passenger engine"
  if-else direction = "east" [
    set current-stop item 0 [eb-sta-num] of stations-here]
  [set current-stop item 0 [wb-sta-num] of stations-here]
  initialize-stops-to-visit
  set initial-time 0
  set stops-visited 0
]
ask trains [set color white
  set size 10
  calculate-time-to-move]

ask trains [
  if train-headway = 15 [
    if train-num = 41 [
      set time-to-move (time-to-move - 180)
    ]
    if train-num = 43 [
      set time-to-move (time-to-move - 660)
    ]
  ]
  if train-headway = 10 [
    if train-num = 43 [
      set time-to-move (time-to-move + 120)
    ]
    if train-num = 45 [
      set time-to-move (time-to-move + 180)
    ]
  ]
  if train-headway = 20 [
    set time-to-move time-to-move
  ]
]
```

```
  ask trains [
    set bike-cap 6
  ]
  ask trains [
    set initial-location [who] of nodes-here
  ]
  ask people [go-dijkstra]

  ask people [find-trax-stations]

  ask people [check-bike-vs-trax]

  ask people [find-direction]

  ask people [find-next-hop-node]

;  ask people [ show still-to-visit ]
;  ask people [ ask location [ set color red] foreach still-to-visit [ ask node ? [ set color
yellow]] ask new-location [ set color orange ]]
  ask people [
    set bike-ticks-to-wait 0
    let cn item 0 [who] of nodes-here
    let nn item 0 still-to-visit
    set bike-ticks-to-wait (([time] of street cn nn) + item 0 [seed-time] of nodes-here)
    set bike-time-to-move (ticks + bike-ticks-to-wait)
  ]

  ask stations [
    if train-headway = 10 [
      set eb-tcounter item 1 counter-list
      set wb-tcounter item 0 counter-list
    ]
    if train-headway = 15 [
      set eb-tcounter item 3 counter-list
      set wb-tcounter item 2 counter-list
    ]
    if train-headway = 20 [
      set eb-tcounter item 5 counter-list
      set wb-tcounter item 4 counter-list
    ]
  ]
  ask stations [
    if station-name = "University Medical Center" [
      if train-headway = 10 or train-headway = 20 [
```

```
      set wait-time 420
     ]
    ]
    if station-name = "Arena" [
     if train-headway = 10 or train-headway = 20 [
      set wait-time 360
     ]
    ]
  ]

  foreach ([who] of stations) [
   ask station ? [
    let number-of-stations max [who] of stations + 1
    set dijkstra-station-distances array:from-list n-values number-of-stations [ number-of-
stations + 10000 ]
    set dijkstra-station-directions array:from-list n-values number-of-stations [nobody]
    if any? nodes-here = True [
     array:set dijkstra-station-distances (item 0 [who] of nodes-here) 0
    ]
   ]
  ]

  let stations-with-nodes NOBODY
  set stations-with-nodes stations with [any? nodes-here = True]
  ask stations-with-nodes [
   go-station-dijkstra
  ]

  find-trax-time
  ask people [
   set rule-follower 0
  ]
  let x people-following-rules
  let y (x * count people)
  ask n-of y people [
   set rule-follower 1
  ]
  ask people [
   set nodes-visited-list []
   set time-visited-list []
   set dist-list []
  ]
  set num-bicyclists 0
  set num-trax-riders 0
  set sum-cyclist-cost []
```

```
  set sum-cyclist-dist[]
  set sum-cyclist-cost fput 0 sum-cyclist-cost
  set sum-cyclist-cost fput 0 sum-cyclist-cost
  set sum-cyclist-dist fput 0 sum-cyclist-dist
  set sum-cyclist-dist fput 0 sum-cyclist-dist
  set sim-num (sim-num + 1)
end

to initialize-distances [cyclist initial-node] ; Initializes the distances before running the
Dijkstra calculations.
 let number-of-nodes max [who] of nodes + 1

 ask cyclist [ set dijkstra-distances array:from-list n-values number-of-nodes [number-of-
nodes + 10000] ];; "infinity"
 ask cyclist [ set dijkstra-directions array:from-list n-values number-of-nodes [nobody] ]
 ask cyclist [ array:set dijkstra-distances initial-node 0 ] ;; set distance to 0 for initial
node
end

to perform-dijkstra [cyclist initial-node nodes-to-visit] ; Used by cyclists to perform the
Dijkstra calculations.

 ask cyclist [ set nodes-visited 0 ]

 initialize-distances cyclist [who] of initial-node

 let visited-set []
 let unvisited-set nodes-to-visit

 let current-node initial-node
 while [count unvisited-set > 0]
 [

  ask current-node
  [
    ask cyclist [ set nodes-visited nodes-visited + 1 ]
    set visited-set fput who visited-set
    set unvisited-set other unvisited-set

    ask out-link-neighbors
    [
      let me who
      let dist-thru-here (array:item [dijkstra-distances] of cyclist [who] of current-node) +
([cost] of in-link-from current-node)
      let dist-thru-to-there array:item [dijkstra-distances] of cyclist who
```

```
      if (dist-thru-here < dist-thru-to-there)
        [ ask cyclist [ array:set dijkstra-distances me dist-thru-here ]
          ask cyclist [ array:set dijkstra-directions me [who] of current-node ] ]
    ]

    set current-node min-one-of unvisited-set [array:item [dijkstra-distances] of cyclist
who]
   ]
 ]
end

to go-dijkstra ; Used by cyclists right before running the Dijkstra calculations.

 let cyclist person item 0 [who] of people-here
 let initial-node node item 0 [who] of nodes-here

 let nodes-to-visit []
 set nodes-to-visit nodes

 perform-dijkstra cyclist initial-node nodes-to-visit
end

to perform-station-dijkstra [trax-station initial-node nodes-to-visit] ; Used by stations to
perfrom Dijkstra calculations.

 ask trax-station [ set station-nodes-visited 0 ]

 let visited-set []
 let unvisited-set nodes-to-visit

 let current-node initial-node
 while [count unvisited-set > 0]
 [

  ask current-node
  [
    ask trax-station [ set station-nodes-visited station-nodes-visited + 1 ]
    set visited-set fput who visited-set
    set unvisited-set other unvisited-set

    ask out-link-neighbors
    [
      let me who
      let dist-thru-here (array:item [dijkstra-station-distances] of trax-station [who] of
current-node) + ([cost] of in-link-from current-node)
```

```
      let dist-thru-to-there array:item [dijkstra-station-distances] of trax-station who
      if (dist-thru-here < dist-thru-to-there)
        [ ask trax-station [ array:set dijkstra-station-distances me dist-thru-here ]
          ask trax-station [ array:set dijkstra-station-directions me [who] of current-node ] ]
    ]

    set current-node min-one-of unvisited-set [array:item [dijkstra-station-distances] of
trax-station who]
   ]
  ]
end

to go-station-dijkstra ; Called by stations right before running the Dijkstra calculations.

  let trax-station station item 0 [who] of stations-here
  let initial-node node item 0 [who] of nodes-here

  let nodes-to-visit []
  set nodes-to-visit nodes

  perform-station-dijkstra trax-station initial-node nodes-to-visit

end

to find-next-hop-node ; Used by cyclists to find the next node in the model they need to
visit on their trip.
  let c-visit-list []

  if (bike-or-trax-path = "Bike") and (bike-type = "Normal") [
    set location one-of nodes-here
    let i [who] of [new-location] of self
    set c-visit-list fput i c-visit-list
    while [ i != [who] of [location] of self ] [
      let j array:item [dijkstra-directions] of self i
      set c-visit-list fput j c-visit-list
      set i j
    ]
    set c-visit-list remove-item 0 c-visit-list
    set still-to-visit c-visit-list
  ]
  if (bike-or-trax-path = "Bike") and (bike-type = "From O Trax") [
    set location one-of nodes-here
    let i [who] of [new-location] of self
    set c-visit-list fput i c-visit-list
    while [ i != [who] of [location] of self ] [
```

```
      let j array:item [dijkstra-from-o-trax-directions] of self i
      set c-visit-list fput j c-visit-list
      set i j
    ]
    set c-visit-list remove-item 0 c-visit-list
    set still-to-visit c-visit-list
  ]
  if bike-or-trax-path = "To Trax" [
    let i item 0 [who] of [closest-o-trax-station] of self
    set c-visit-list fput i c-visit-list
    while [ i != [who] of [location] of self ] [
      let j array:item [dijkstra-directions] of self i
      set c-visit-list fput j c-visit-list
      set i j
    ]
    set c-visit-list remove-item 0 c-visit-list
    set still-to-visit c-visit-list
  ]
  if bike-or-trax-path = "From Trax" [
    let i [who] of [new-location] of self
    set c-visit-list fput i c-visit-list
    while [ i != [who] of [location] of self ] [
      let j array:item [dijkstra-from-trax-directions] of self i
      set c-visit-list fput j c-visit-list
      set i j
    ]
    set c-visit-list remove-item 0 c-visit-list
    set still-to-visit c-visit-list
  ]
end

to calculate-bike-time-to-move ; Used by cyclists to determine when they can move next.
  set bike-ticks-to-wait 0
  let cn item 0 [who] of nodes-here
  let nn item 0 still-to-visit
  set bike-ticks-to-wait ([time] of street cn nn)
  set bike-time-to-move (ticks + bike-ticks-to-wait)
end

to find-trax-stations ; Used by cyclists to find the closest TRAX stations to the origin and
destination nodes.
  let my-o-trax NOBODY
  let node-o-trax [closest-station] of nodes-here
  set my-o-trax nodes with [([closest-station] of nodes-here = node-o-trax) and (is-station
= "Yes")]
```

```
  set closest-o-trax-station my-o-trax
  let my-d-trax NOBODY
  let node-d-trax [closest-station] of new-location
  set my-d-trax nodes with [(closest-station = node-d-trax) and (is-station = "Yes")]
  set closest-d-trax-station my-d-trax
end

to check-bike-vs-trax ; Used by cyclists to initially set their paths as cycle-only or move
toward a TRAX station.
  let a [who] of new-location
  let b item 0 [who] of closest-o-trax-station
  let bike-cost array:item dijkstra-distances a
  let trax-cost array:item dijkstra-distances b
  if-else ((trax-cost / bike-cost) < 0.5) [
    set bike-or-trax-path "To Trax"
  ]
  [ set bike-or-trax-path "Bike"
    set bike-type "Normal"
  ]
  if [closest-station] of closest-o-trax-station = [closest-station] of closest-d-trax-station [
    set bike-or-trax-path "Bike"
    set bike-type "Normal"
  ]
  set location one-of nodes-here
end

to find-direction ; Used by cyclists to determine their trip direction (east / west).
  if-else ([xcor] of new-location - [xcor] of location) > 0 [
    set bike-direction "east"
  ]
  [ set bike-direction "west"
  ]
end

to reset-station-counter ; Resets the time counter at the TRAX stations alerting cyclists
when the next train will arrive.
  if train-headway = 10 [
    if-else direction = "east" [
      ask stations-here [
        set eb-tcounter 600
      ]
    ]
    [ ask stations-here [
        set wb-tcounter 600
    ]
```

```
    ]
  ]
  if train-headway = 15 [
    if-else direction = "east" [
      ask stations-here [
        set eb-tcounter 900
      ]
    ]
    [ ask stations-here [
        set wb-tcounter 900
    ]
    ]
  ]
  if train-headway = 20 [
    if-else direction = "east" [
      ask stations-here [
        set eb-tcounter 1200
      ]
    ]
    [ ask stations-here [
        set wb-tcounter 1200
    ]
    ]
  ]
end

to re-evaluate-path ; Used by cyclists to determine whether a they should wait for the
train or continue cycling.
  find-from-o-trax-path
  calculate-trax-path-cost
  let x [who] of new-location
  let y array:item dijkstra-from-o-trax-distances x
  type "Bike path cost is " type y
  let tmax 0
  let cost-difference (y - trax-path-cost)
  if-else cost-difference > 0 [
    set bike-or-trax-path "From Trax"
    set bike-type "Normal"
  ]
  [ set bike-or-trax-path "Bike"
    set bike-type "From O Trax"
  ]
  if train-headway = 10 [
    set tmax 600 ]
  if train-headway = 15 [
```

```
    set tmax 900 ]
  if train-headway = 20 [
    set tmax 1200 ]
  if bike-direction = "east" [
    if item 0 [eb-tcounter] of stations-here > (tmax * bike-wait-time) [
      set bike-or-trax-path "Bike"
      set bike-type "From O Trax"
    ]
  ]
  if bike-direction = "west" [
    if item 0 [wb-tcounter] of stations-here > (tmax * bike-wait-time) [
      set bike-or-trax-path "Bike"
      set bike-type "From O Trax"
    ]
  ]
  if bike-or-trax-path = "Bike" [
    find-next-hop-node
    calculate-bike-time-to-move
  ]
  if bike-or-trax-path = "From Trax" [
    board-trains
    set riding-train? False
  ]
end

to ride-trains ; Used by the cyclists to ride the train from one TRAX station to the next.
  if my-train = NOBODY [
    board-trains
  ]
  let my-station NOBODY
  let s 0
  ask closest-d-trax-station [
    set s item 0 [who] of stations-here
  ]
  set my-station station s
  if one-of stations-here = my-station [
    type "cyclist " type who type " is getting off the train"
    set my-train NOBODY
    de-board-trains
  ]
  if my-train != NOBODY [
    move-to train my-train
  ]
end
```

```
to de-board-trains ; Used by cyclists at the egress TRAX station to get off of the train.
  set location one-of nodes-here
;  type "location set"
  set hidden? False
;  type " cyclist visible"
  set riding-train? False
;  type " not riding train"
  find-from-d-trax-path
;  type " path found"
  find-next-hop-node
;  type " next hop node found"
  calculate-bike-time-to-move
;  type " time to move found"
  ask trains-here [
    set bike-cap (bike-cap + 1)
  ]
;  type " train bicycle capacity set"
end

to find-from-d-trax-path ; Used by cyclists to find a path to the destination from the
TRAX station.
  let my-from-trax-list []
  let my-station nobody
  let s first [who] of stations-here
  set my-station station s
  ask my-station [
    set my-from-trax-list array:to-list dijkstra-station-directions
  ]
  set dijkstra-from-trax-directions array:from-list my-from-trax-list
end

to find-from-o-trax-path ; Used by cyclists to find a path to the destination from the
origin TRAX station.
  let my-from-o-trax-list []
  let my-from-o-trax-distances []
  let my-station nobody
  let s first [who] of stations-here
  set my-station station s
  ask my-station [
    set my-from-o-trax-list array:to-list dijkstra-station-directions
    set my-from-o-trax-distances array:to-list dijkstra-station-distances
  ]
  set dijkstra-from-o-trax-directions array:from-list my-from-o-trax-list
  set dijkstra-from-o-trax-distances array:from-list my-from-o-trax-distances
end
```

```
to go-trax-dijkstra

  let cyclist person item 0 [who] of people-here
  let initial-node node first [who] of closest-d-trax-station

  let nodes-to-visit [] ;; this is the list of nodes yet to have been visited
  set nodes-to-visit nodes

  perform-dijkstra cyclist initial-node nodes-to-visit
end

to board-trains ; Used by cyclists to determine which train to wait for and to board the
train when it arrives.
  if one-of nodes-here != closest-d-trax-station [
    set hidden? True
  ]
  let my-location NOBODY
  let my-direction NOBODY
  set my-train NOBODY
  set my-direction bike-direction
  let my-train-who NOBODY
  if any? trains-here [
    foreach ([who] of trains) [
      ask train ? [
        if direction = my-direction [
          set my-train-who [who] of self
        ]
      ]
    ]
  ]
  set my-train my-train-who
  if any? trains-here [
    let x my-train
    if rule-follower = 1 [
      if [bike-cap] of train x > 0 [
        move-to train x
        set riding-train? True
        ask train x [
          set bike-cap (bike-cap - 1)
        ]
      ]
      if [bike-cap] of train x = 0 [
        board-trains
      ]
```

```
    ]
    if rule-follower = 0 [
      move-to train x
      set riding-train? True
      ask train x [
        set bike-cap (bike-cap - 1)
      ]
    ]
  ]
end

to calculate-trax-path-cost ; Used by cyclists to determine the cost of a TRAX-based path.
  if-else bike-direction = "east" [
    let i item 0 [eb-sta-num] of stations-here
    let j 0
    let my-eb-sta-num 0
    ask closest-d-trax-station [
      ask stations-here [
        set my-eb-sta-num eb-sta-num
      ]
    ]
    set j my-eb-sta-num
    let k 0
    while [i < j] [
      set k k + (array:item eb-trax-cost i)
      set i i + 1
    ]
    let x array:item dijkstra-distances (item 0 [who] of closest-d-trax-station)
    set trax-path-cost k + x
    type " Trax path cost is " type trax-path-cost
  ]
  [ let l item 0 [wb-sta-num] of stations-here
    let m 0
    let my-wb-sta-num 0
    ask closest-d-trax-station [
      ask stations-here [
        set my-wb-sta-num wb-sta-num
      ]
    ]
    set m my-wb-sta-num
    let n 0
    while [l < m] [
      set n n + (array:item wb-trax-cost l)
      set l l + 1
    ]
```

```
    let y array:item dijkstra-distances (item 0 [who] of closest-d-trax-station)
    set trax-path-cost n + y
    type " Trax path cost is" type trax-path-cost
   ]
end

to find-trax-time ; Used by cyclists to set the TRAX-cost and TRAX-distance arrays.
  set wb-trax-cost array:from-list n-values (count stations + 1) [count stations]
  set eb-trax-cost array:from-list n-values (count stations + 1) [count stations]
  set wb-trax-dist array:from-list n-values (count stations + 1) [count stations]
  set eb-trax-dist array:from-list n-values (count stations + 1) [count stations]
  foreach ([who] of stations) [
    ask station ? [
      let i [eb-sta-num] of self
      if i < 31 [
        let j (i + 1)
        let k [who] of self
        let l item 0 [who] of stations with [eb-sta-num = j]
        let m [cost] of trax_line k l
        let x [dist] of trax_line k l
        array:set eb-trax-cost i m
        array:set eb-trax-dist i x
        ]
      let n [wb-sta-num] of self
      if n < 31 [
        let o (n + 1)
        let p [who] of self
        let q item 0 [who] of stations with [wb-sta-num = o]
        let r [cost] of trax_line p q
        let y [dist] of trax_line p q
        array:set wb-trax-cost n r
        array:set wb-trax-dist n y
      ]
     ]
    ]
end

to find-trax-dist-cost ; Used by cyclists to determine the cost of taking a TRAX-based
path.
  let a 0
  let b 0
  set trax-ride-cost 0
  set trax-dist 0
  if-else bike-direction = "east" [
    ask closest-o-trax-station [
```

```
      set a item 0 [eb-sta-num] of stations-here
    ]
  ]
   [ ask closest-o-trax-station [
       set a item 0 [wb-sta-num] of stations-here
  ]
   ]
  if-else bike-direction = "east" [
    ask closest-d-trax-station [
      set b item 0 [eb-sta-num] of stations-here
    ]
  ]
   [ ask closest-d-trax-station [
       set b item 0 [wb-sta-num] of stations-here
  ]
   ]
  if-else bike-direction = "east" [
    while [a < b] [
      set trax-ride-cost (trax-ride-cost + array:item eb-trax-cost a )
      set trax-dist (trax-dist + array:item eb-trax-dist a)
      set a (a + 1)
    ]
  ]
  [ while [a < b] [
      set trax-ride-cost (trax-ride-cost + array:item wb-trax-cost a)
      set trax-dist (trax-dist + array:item wb-trax-dist a)
      set a (a + 1)
  ]
  ]
end

to go ; The procedure controlling the movement of cyclists and trains.
  tick
  foreach ([who] of people) [
    ask person ? [
    let next-hop-node NOBODY
    let cn NOBODY
    let nn NOBODY
    if bike-or-trax-path = "Bike" [
      if length still-to-visit > 0 [
        set next-hop-node node item 0 still-to-visit
        set cn item 0 [who] of nodes-here
        set nn item 0 still-to-visit
      ]
      if ticks = bike-time-to-move [
```

```
    move-to next-hop-node
    set nodes-visited-list fput item 0 [id] of nodes-here nodes-visited-list
    set time-visited-list fput ticks time-visited-list
    set dist-list fput [dist] of street cn nn dist-list
    set still-to-visit remove-item 0 still-to-visit
    type "cyclist " type who type " still has to visit: " print still-to-visit
    if length still-to-visit > 0 [
      calculate-bike-time-to-move
    ]
  ]
  if length still-to-visit = 0 [
    type "Destination reached for cyclist " type who
    set num-bicyclists (num-bicyclists + 1)
    let dist-sum sum dist-list
    set sum-cyclist-dist fput dist-sum sum-cyclist-dist
    set sum-cyclist-cost remove 0 sum-cyclist-cost
    set sum-cyclist-dist remove 0 sum-cyclist-dist
    set nodes-visited-list reverse nodes-visited-list
    set time-visited-list reverse time-visited-list
    set dist-list reverse dist-list
    output-show nodes-visited-list
    output-show time-visited-list
    output-show dist-list
    die
  ]
]
if bike-or-trax-path = "To Trax" [
  if length still-to-visit > 0 [
    set next-hop-node node item 0 still-to-visit
    set cn item 0 [who] of nodes-here
    set nn item 0 still-to-visit
  ]
  if ticks = bike-time-to-move [
    move-to next-hop-node
    set nodes-visited-list fput item 0 [id] of nodes-here nodes-visited-list
    set time-visited-list fput ticks time-visited-list
    set dist-list fput [dist] of street cn nn dist-list
    set still-to-visit remove-item 0 still-to-visit
    type "cyclist " type who type " still has to visit: " print still-to-visit
    if length still-to-visit > 0 [
      calculate-bike-time-to-move
    ]
  ]
  if length still-to-visit = 0 [
    let time-to-next-train 0
```

```
    if-else bike-direction = "east" [
      set time-to-next-train [eb-tcounter] of stations-here
    ]
    [ set time-to-next-train [wb-tcounter] of stations-here
    ]
    re-evaluate-path
  ]
]
if bike-or-trax-path = "From Trax" [
  if (riding-train? = False) and (length still-to-visit = 0) [
    board-trains
  ]
  if (riding-train? = False) and (length still-to-visit > 0) [
    set next-hop-node node item 0 still-to-visit
    set cn item 0 [who] of nodes-here
    set nn item 0 still-to-visit
    if ticks = bike-time-to-move [
      move-to next-hop-node
      set nodes-visited-list fput item 0 [id] of nodes-here nodes-visited-list
      set time-visited-list fput ticks time-visited-list
      set dist-list fput [dist] of street cn nn dist-list
      set still-to-visit remove-item 0 still-to-visit
      type "cyclist " type who type " still has to visit: " print still-to-visit
      if length still-to-visit > 0 [
        calculate-bike-time-to-move
      ]
    if length still-to-visit = 0 [
      type "Destination reached for cyclist " type who type " who rode TRAX"
      set num-trax-riders (num-trax-riders + 1)
      find-trax-dist-cost
      set dist-list fput trax-dist dist-list
      let dist-sum sum dist-list
      set sum-cyclist-dist fput dist-sum sum-cyclist-dist
      set sum-cyclist-cost remove 0 sum-cyclist-cost
      set sum-cyclist-dist remove 0 sum-cyclist-dist
      set nodes-visited-list reverse nodes-visited-list
      set time-visited-list reverse time-visited-list
      set dist-list reverse dist-list
      output-show nodes-visited-list
      output-show time-visited-list
      output-show dist-list
      die
    ]
    ]
  ]
```

```
      if (riding-train? = True) [
        ride-trains
      ]
    ]
  ]
  ]
  if count people = 0 [stop]
  ask stations [
    set eb-tcounter eb-tcounter - 1
    set wb-tcounter wb-tcounter - 1
  ]
  foreach ([who] of trains) [
    ask train ? [
    let next-hop-station NOBODY
    if length stops-to-visit > 0 [
      if-else direction = "east" [
        let t array:item eb-sta-order (item 0 stops-to-visit)
        set next-hop-station station t
        ]
      [ let w array:item wb-sta-order (item 0 stops-to-visit)
        set next-hop-station station w
      ]
    ]
    if ticks = time-to-move [
      move-to next-hop-station
      set stops-visited (stops-visited + 1)
      set stops-to-visit remove-item 0 stops-to-visit
      reset-station-counter

      if (length stops-to-visit) = 0 [
        ask train ? [ change-directions
          reset-station-counter ]
      ]
      ask train ? [
        calculate-time-to-move
      ]
      ]
  ] ]
end
```

LITERATURE CITED

Ben-Akiva, M.E. and S.R. Lerman (1985) *Discrete choice analysis: theory and application to travel demand*, MIT Press, Cambridge.

Bracher, T. (2000) "Demand characteristics and co-operation strategies for the bicycle and railway transport chain". *World Transport Policy & Practice*, Vol. 6. No. 3, 18-24.

Brown, D.G., R. Riolo, D.T. Robinson, M. North, and W. Rand. (2005) "Spatial process and data models: toward integration of agent-based models and GIS". *Journal of Geographical Systems Vol. 7*, 25-47.

Brownstone, D. (2001) "Discrete choice modeling for transportation", in D.A. Hensher (ed.) *Travel behavior research: the leading edge*, 97–124, Elsevier, Oxford.

Cervero, R. (2006) "Alternative approaches to modeling the travel-demand impacts of smart growth". *Journal of the American Planning Association Vol. 72, No. 3*, 285-295.

Cormen, T.H., C.E. Leiserson, R.L. Rivest and C. Stein. (2001) *Introduction to algorithms: second edition*, MIT Press.

Davidsson, P., L. Henesey, L. Ramstedt, J. Tornquist and F. Wernstedt. (2005) "An analysis of agent-based approaches to transport logistics". *Transportation Research Part C Vol. 13*, 255-271.

Dijkstra, E.W. (1959) "A note on a two problems in connection with graphs". *Numerische Mathematik*, Vol. 1, 269-271.

Dill, J. and T. Carr. (2003) "Bicycle commuting and facilities in major U.S. cities: if you build them, commuters will use them," *Transportation Research Record No. 1828*, 116-123.

Dill, J. and K. Voros. (2007) "Factors affecting bicycling demand: initial survey findings from the Portland region." *Transportation Research Record: Journal of the Transportation Research Board No. 2031,* 9-17.

Dixon, L.B. (1996) "Bicycle and pedestrian level of service performance measures and standards for congestion management systems". *Transportation Research Record No. 1538,* 1-9.

Flake, G.W. (1998) *The computational beauty of nature*, MIT Press.

Gimblett, H.R. (2002) “Integrating geographic information systems and agent-based technologies for modeling and simulating social and ecological process”. In *Integrating geographic information systems and agent-based modeling technologies for simulating social and ecological processes*, ed. H.R. Gimblett, 1-20. Oxford: Oxford University Press.

Givoni, M. and P. Rietveld. (2006) "Access to railway stations in the Netherlands". *European Regional Science Association*. ERSA conference papers ersa06p506.

Guttenplan, M., B.W. Landis, L. Crider and D.S. McLeod. (2001) “Multimodal level of service (LOS) analysis at a planning level”. *Transportation Research Record No. 1776,* 151-158.

Israelsen, T., and R. D. Frederiksen. 2005. The use of GIS in transport modeling. In *GIS, Spatial Analysis and Modeling*, ed. D.G. Maguire, M. Batty, and M. Goodchild, 265-288. Redlands, CA: ESRI Press.

Keijer, M.J.N. and P. Rietveld. (2000) “How do people get to the railway station? The Dutch experience”. *Transport Planning and Technology*, Vol. 23, 215-235.

Kikuchi, S., J. Rhee and D. Teodorovic. (2002) “Applicability of an agent-based modeling concept to modeling of transportation phenomena”. *Yugoslav Journal of Operations Research Vol. 12, No. 2*, 141-156.

Krizek, K.J. (2003) “Neighborhood services, trip purpose, and tour-based travel”. *Transportation*, Vol. 30, Issue 4, 387-410.

Kuby, M., A. Barranda and C. Upchurch. (2004) “Factors influencing light-rail station boardings in the United States”. *Transportation Research Part A Vol. 38,* 223-247.

Landis, B.W., V.R. Vattikuti, and M.T. Brannick. (1997) “Real time human perceptions: toward a bicycle level of service”. *Transportation Research Record No. 1578*, 119-126.

Lane, C., M. DiCarlantonio, and L. Usvyat. (2006) “Sketch models to forecast commuter and light rail ridership”. *Transportation Research Record No. 1986*, 198-210.

League of American Bicyclists. (2011) List of bicycle-friendly communities. http://www.bikeleague.org/programs/bicyclefriendlyamerica/communities/.

Levy, J.M. (2000) *Contemporary urban planning, 5th ed.*, Prentice-Hall.

Lu, Y., K. Kawamura and M.L. Zellner. (2008) “Exploring the influence of urban form on work travel behavior with agent-based modeling”. *Transportation Research Record No. 2082*, 132-140.

Lythgoe, W.F., M. Wardman, and J.P. Toner. (2004) "Enhancing rail passenger demand models to examine station choice and access to the rail network". *AET European Transport Conference,* PTRC, London.

Martens, K. (2004) "The bicycle as a feedering mode: experiences from three European countries". *Transportation Research Part D,* Vol. 9 No. 4, 281-294.

Martens, K. (2007) "Promoting bike-and-ride: the Dutch experience". *Transportation Research Part A*, Vol. 41, 326-338.

McFadden, D. (1974) "The measurement of urban travel demand". *Journal of Public Economics*, Vol. 3, 303-328.

McLeod, D.S. (2000) "Multimodal arterial level of service". *Transportation Research E-Circular E-C018: 4th International Symposium on Highway Capacity,* 221-233.

McNally, M.G. (1997) "An activity-based microsimulation model of travel demand forecasting", in D.F. Ettema and H.J.P. Timmermans (eds.) *Activity-Based Approaches to Travel Demand Forecasting*, 37-54. Pergamon Press, Oxford.

Miller, H.J. and S-L Shaw. (2001) *Geographic information systems for transportation: principles and applications*, Oxford University Press.

Miller, H.J. (2007) "Geocomputation," in A. S. Fotheringham and P. A. Rogerson (eds.) *Handbook of spatial analysis*, Sage Publications.

Molin, E. and M. van Gelder. (2008) "Freeway access to public transport". *Transportation Research Record No. 2076*, 106-113.

Moudon, A.V. and C. Lee. (2003) "Walking and bicycling: an evaluation of environmental audit instruments". *American Journal of Health Promotion* Vol. 18. No. 1, 21-37.

Moudon, A.V., C. Lee, A.D. Cheadle, C.W. Collier, D. Johnson, T.L. Schmid, and R.D. Weather. (2005) "Cycling and the built environment: a U.S. perspective". *Transportation Research Part D Vol. 10, 245-261.*

Noland, R.B. and H. Kunreuther. (1995) "Short-run and long-run policies for increasing bicycle transportation for daily commuter trips". *Transport Policy*, Vol. 2 No. 1, 67-79.

Replogle, M. (1992) "Bicycle access to public transportation: learning from abroad". *Institute of Transportation Engineers Journal*, Vol. 62 No. 12, 17-21.

Rietveld, P. (2000) "The accessibility of railway stations: the role of the bicycle in the Netherlands". *Transportation Research Part D*, Vol. 5, 71-75.

Smillie, E. (2009) "Sorry Portland: a primer on the best burgeoning bike scenes in North America". *Good Magazine Issue 015*, 52-55.

Sobel, K.L. (1980) "Travel demand forecasting by using the nested multinomial logit model". *Transportation Research Record No. 775*, 48-55.

Southworth. F. (1985) "Multi-destination, multi-purpose trip chaining and its implications for locational accessibility: a simulation approach". *Papers of the Regional Science Association*, Vol. 57. Issue 1, 107-123.

Teahan, W. J. (2010) Being Kevin Bacon NetLogo model. Artificial Intelligence. *Ventus Publishing Aps.*

Topp, H.H. (1999) "Innovations in tram and light rail systems". *Proceedings of Institution of Mechanical Engineers*, Vol. 213, Part F, 133-141.

Toroczkai, Z. and H. Guclu. (2007) "Proximity networks and epidemics". *Physica A: Statistical Methods and it Applications*, Vol. 378. No. 1, 68-75.

Transit Cooperative Research Program. (1994) "Synthesis 4: integration of bicycles and transit". *Transportation Research Board*: Washington, D.C.

Transit Cooperative Research Program. (2005) "Synthesis 62: integration of bicycles and transit". *Transportation Research Board*: Washington, D.C.

Turner, S., A. Hottenstein, and G. Shunk. (1997) "Bicycle and pedestrian travel demand forecasting: a literature review". *Texas Transportation Institute Report 1723-1*, 1-44.

Wardman, M. and J. Tyler. (2000) "Rail network accessibility and the demand for inter-urban rail travel". *Transport Reviews*, Vol. 20 No. 1, 3-24.

Wilensky, U. (2005) NetLogo Small Worlds model. http://ccl.northwestern.edu/netlogo/models/SmallWorlds. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Wilensky, U. (1999). NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Worboys, M. and M. Duckham. (2004) *GIS: a computing perspective*, CRC Press.

Zhang, L. and D. Levinson. (2004) "An agent-based approach to travel demand modeling: an exploratory analysis". *Transportation Research Record No. 1898*, 28-38.