EXAMINING COST ASPECTS OF SHARED AUTONOMOUS VEHICLES (SAV) OPERATED AS MOBILITY AS A SERVICE (MAAS)

A Thesis

by

SAIPRANEETH DEVUNURI

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Mark Burris
Co-Chair of Committee,	Sivakumar Rathinam
Committee Member,	Alireza Talebpour
Head of Department,	Robin Autenrieth

August 2020

Major Subject: Civil Engineering

ABSTRACT

Autonomous vehicle (AV) technology is rapidly developing and is predicted to have a significant impact on travel. One application of AV technology is Shared Autonomous Vehicles (SAV), where a single AV is shared amongst several users. SAV may prove to be more cost-effective than ownership of personal vehicles. This could imply an increasing demand for ride-sharing services and autonomous taxis (aTaxi) and increasing use of Mobility as a Service (MaaS). Moreover, Shared Autonomous Electric Vehicles (SAEV) reduce two significant cost factors for ridesharing, driver, and fuel. Thus, SAEV could dramatically change the economics of offering ridesharing or aTaxi services and potentially compete with existing TNCs (Transportation Network Companies) and vehicle ownership itself. Companies like Uber, Tesla, and Lyft are looking into various business models to capture this market. One such model is to incorporate a subscription-based pricing model, similar to Spotify or Netflix. This research evaluated a potential pricing structure for an autonomous ridesharing system under subscription-based pricing and compare it with existing ridesharing systems based on pay per ride and personal vehicles. The results show that a monthly fee of $170 can be charged per traveler with a per-mile cost varying between $0.31 and $0.71.

DEDICATION

I dedicate this research to my parents; whose persistent support was indispensable in pursuing my passion. A special mention to my sister who revered my success and celebrated it with me.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Mark Burris who helped me come up with this research idea and guided me a lot throughout this research. I thank Dr. Sivakumar Rathinam and Dr. Alireza Talebpour for serving as the Co-Chair and Member of my thesis committee. They provided me valuable inputs and added different perspectives to this research.

I thank the contributions of Ankit Jhamb, who had a key role in coming up with the research idea and worked on the financial aspects of the research. I would also like to thank Dr. Hao Pang for providing me the necessary data required to conduct this research.

I would like to thank my colleagues who helped me nurture the research culture and understand concepts better. Finally, I thank my roommates who brainstormed with me on several ideas and were part of everything big and small, good and bad in my journey at Texas A&M University.

CONTRIBUTORS AND FUNDING SOURCES

This study was supervised by a thesis committee including Dr. Mark Burris and Dr. Alireza Talebpour of the Department of Civil Engineering, and Dr. Sivakumar Rathinam of the Department of Mechanical Engineering. Certain portions of the Financial Analysis were performed by Ankit Jhamb of the Department of Civil Engineering. The data collection and a few pre-processing tasks were conducted by Dr. Hao Pang at Texas A&M Transportation Institute (TTI), Austin. There was no funding source for this research

NOMENCLATURE

SAV	Shared Autonomous Vehicle
SAEV	Shared Autonomous Electric Vehicle
TNC	Transportation Network Companies
MaaS	Mobility as A Service
AV	Autonomous Vehicle
AEV	Autonomous Electric Vehicle
NHTS	National Household Travel Survey
CAMPO	Capital Area Metropolitan Planning Organization
TAZ	Traffic Analysis Zone
O-D	Origin – Destination
NCS	Nearest Charging Stations
NNCS	Next Nearest Charging Stations
AAA	American Automobile Association
AM	Ante Meridiem
MD	Mid-Day
PM	Post Meridiem
NT	Night
VO	Vehicle Ownership

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

1. INTRODUCTION

1.1. Overview

Technology has a great deal of impact on our travel behavior. Even before a trip starts, technology can help travelers plan their trip and a whole new set of technologies help in completing the trip. Autonomous Vehicle (AV) technology may have a significant impact on travel and may offer unprecedented opportunities for businesses. The autonomous vehicle (AV) industry is developing rapidly and is estimated to reach $173 billion by 2023 (HTF Market Intelligence, 2019). The anticipated exponential growth in adopting autonomous vehicles by the industry as well as the general public is due to many advantages offered by these vehicles, mainly improving safety and reducing congestion. As per the 2015 Summary of Motor Vehicle Crashes (Final Edition) published by NHTSA, human error is responsible for over 90 percent of vehicle crashes. It is expected that fully autonomous vehicles will improve safety on the road by eliminating human driving. Also, AVs will add the capability to carry children and elderly people to the destination in a safe manner increasing vehicle trips making the market open for most age groups irrespective of whether the user can drive or not. They may improve the capacity of roadways due to the vehicle to vehicle/infrastructure coordination.

Meanwhile, the market for ride-sharing services is predicted to double in revenue from 2017-2023. These services affect people's choice to use cars as a service rather than owning personal vehicles. However, for average travelers owning a personal vehicle is economically more beneficial in most cases. With AVs, one of the most substantial costs

of ridesharing, the driver, is removed. About two-thirds of the revenue generated by Transportation Network Companies (TNCs) is used for paying the drivers (Ulama, 2016). Therefore, AVs may further boost the ridesharing market. Various TNCs are experimenting in this field. For example, Google (Waymo) has started using self-driving cars for its ride-sharing service in Phoenix, AZ (https://waymo.com/journey/).

Vehicle ownership is considered a major reason for people to not look for alternative modes of travel. With the introduction of AVs, Schoettle and Sivak (2015) indicated that the average household vehicle ownership as per NHTS (National Household Travel Survey) could be substantially reduced by roughly 40% to 1.2 vehicles per household. However, Freedman (2018) states that currently, Autonomous vehicles (AVs) lifetime costs exceed the cost of human-driven personal cars by 49%, but this is expected to come down to 6% considering 5-year Moore's law prediction. Therefore, in the near future, owning an AV will likely be very expensive, due to the high purchase and maintenance cost. Hence, it makes the use of AVs as taxis or SAVs more economical and more likely. Fagnant et al. (2015) ran a simulation based on travel demand for the city of Austin in Texas, whose results claimed that 1 SAV could alone complete as many intra-urban trips as 9 conventional vehicles.

Mobility as a Service (MaaS) will play a vital role in introducing this technology to the general public before many individuals own an AV. AARP Public Policy Institute (2018) describes MaaS as "a shift away from personally-owned modes of transportation (i.e., car ownership) and towards mobility solutions consumed as a service." One advantage of MaaS would be that users could choose the best mode of travel for them in

terms of cost and travel time for that particular trip. Juniper and Moovel (2017) claimed that subscription models would become the basis of MaaS and introduced various business models with a collection of pricing schemes.

1.2. Problem Statement

Predictions from various sources (IHS Markit, Statista) indicate that AV and ridesharing market share are set to rise rapidly soon. This, in turn, would possibly increase the autonomous taxis (aTaxi) services market share as well. However, this rise can only be sustained if people shift from personally owned vehicles towards MaaS. This change could be fostered by subscription-based pricing schemes which would retain a customer in business for longer duration thereby ensuring constant demand for service. This research examines the economic viability of SAV as MaaS for both business operators and riders.

1.3. Research Objectives

Collectively, various research studies have investigated the concept of Shared Autonomous Vehicles (SAV) and worked on developing a ride-sharing system. However, very few research studies have considered the concept of subscription-based ridesharing and worked on its financial viability. Moreover, there are very few studies that help users determine the best mode of travel between pay-per ride ridesharing, subscription-based ridesharing, and personal vehicle, based on their travel needs. The current research aims to bridge these gaps by achieving the following objectives:

- Examine cost aspects of SAV to operate in ridesharing mode
    - Build a flexible simulation of the ridesharing environment

        - Trip Generation
        - Trip Processing
        - Charging
    - Determine the fleet size for a given demand and Market Penetration Rate (MPR)
    - Evaluate the fixed and operating costs for a fleet size
- Formulate different subscription-based pricing schemes
    - Pricing scheme to break-even
    - Pricing scheme to compete with current TNCs (Uber, Lyft, etc.)
    - Pricing scheme to compete with vehicle ownership
- Evaluate the best economic strategy for consumers based on monthly travel needs
    - Compare TNC pricing with vehicle ownership costs
    - Compare SAV pricing with vehicle ownership costs

1.4. Research Benefits

This research would be most beneficial to ridesharing companies with AVs such as Uber, Lyft, Waymo, and Cruise that are looking for innovative pricing schemes to retain their customer base. Also, the research would serve as a way to determine their operational AV fleet size for any city/town of interest. The detailed cost analysis and considerations will help estimate the preliminary costs for setting up the SAV system and gauging the extent of profits in the system. The research also helps the general public to develop a deeper understanding of the cost of ridesharing and make rational decisions in choosing these services.

The next chapter presents insights into previous literature in the area of ridesharing, especially with SAVs.

2. LITERATURE REVIEW

2.1. Overview

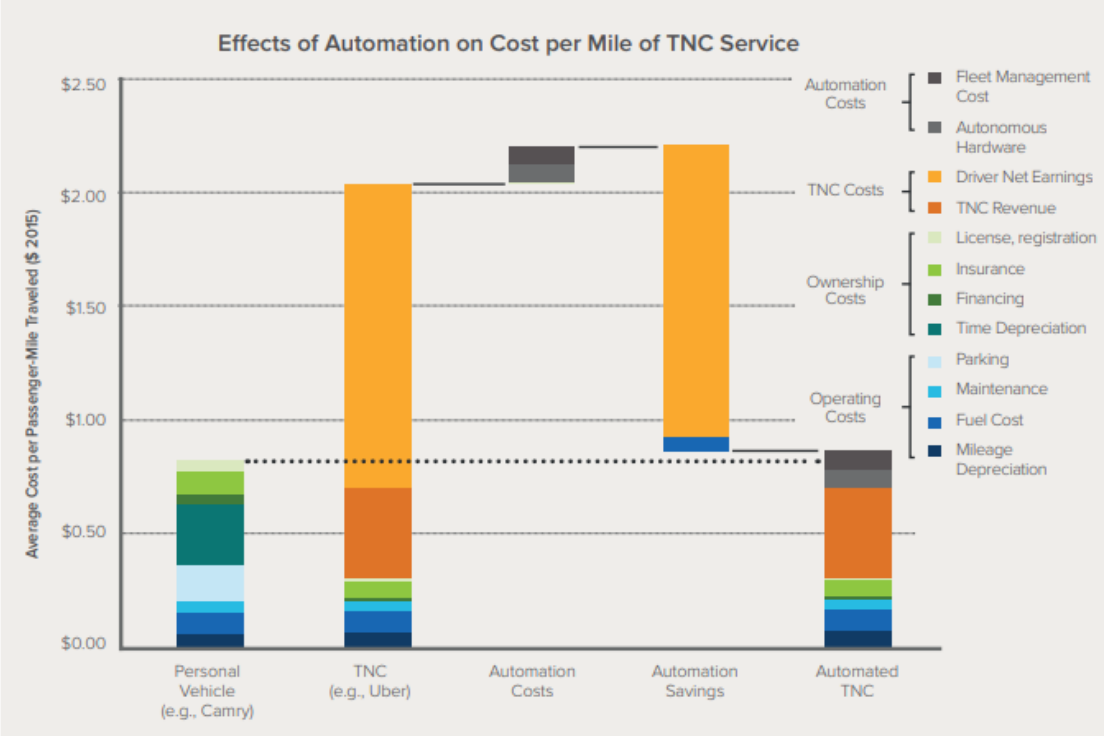
Evaluating the potential cost and revenues from ridesharing services is key for companies looking to enter the ridesharing market. Several researchers have attempted to simulate ride-sharing systems with and without AVs to evaluate pricing, improve operational efficiency, and improve understanding of the ride-sharing systems. However, there is limited public research into the potential costs and revenues from SAVs used as MaaS with subscription-based pricing. There is likely considerable research done by current and potential ridesharing companies that are not publicly available. The financial viability of SAV with subscription pricing depends on the fixed costs which include vehicle ownership and infrastructure (charging stations) costs and operational costs which include fleet management, fuel, and other miscellaneous costs.

The cost of owning an Electric Vehicle (EV) has been decreasing since 2005 and is expected to continue to decline (Nykvist and Nilsson, 2015). The per-mile cost to power an electric-powered AV is lower compared to gas-powered AV. Also, unlike gas-powered vehicles, most EVs come with drive-by-wire systems, which help in running the cars autonomously without additional equipment. Therefore, AEV (Autonomous Electric Vehicle) may be more economical and viable than gas-powered AVs. However, owning any type of AV may still be years away as AVs are expected to be very soon. Therefore, it is more likely that individuals use SAVs as part of a ridesharing service.

2.2. Ridesharing Costs

To better estimate all the potential costs of a ridesharing service many researchers have modeled those services. Loeb et al. (2019) simulated both the electric and hybrid SAV (Shared Autonomous Vehicles) used as a ride-sharing fleet. They used tour patterns developed based on NHTS and US census data in Austin, Texas, and simulated ridesharing using MATSim. By comparing the cost per mile for 6 types of vehicle fleets (Gasoline Hybrid-electric SAEV, combinations of short and long-range SAEVs with slow charge and fast charge and long-range SAEVs having reduced fleet) they found fast charge long-range SAEVs fleet to be the most advantageous EV option. This paper further developed this idea, by running simulations of ridesharing with a fleet of Tesla Model 3, a fast charge long-range SAEV (220 mileage) vehicle fleet, across the city of Austin, Texas.

Similarly, a study by Aditi et al. (2017) examined SAV (Hybrid electric vehicle with a 60-mile battery range) in Ann Arbor, Michigan, and compared it to other alternatives. They found that public transport combined with first and last-mile connectivity using AVs promotes MaaS and has significant benefits over other alternatives, including energy savings up to 37 percent. However, to truly promote MaaS, the cost of the ridesharing services should be comparable to that of owning a personal vehicle. Johnston and Walker (2017) state that the average cost per mile of hiring Uber or Lyft comes out to be twice or thrice of what it cost to own and operate a personal vehicle. The study from Charlie and Jonathan (2017) show that AVs used for mobility service would cost roughly like what it cost for operating and owning a sedan like Camry (see Figure 1).



Reprinted from Charlie and Jonathan, 2017

Figure 1 Effects of Automation on the Cost of TNC Services

2.3. Acceptance of SAVs

The acceptance of SAVs will also increase due to the fact they will likely be cheaper than current TNCs. A study of consumer behavior in Austin, Texas by Bansal (2016) state they noticed a jump of 26% in the number of residents who would choose to travel via SAVs at least once a week due to lower prices as compared to the current rates of using conventional vehicles in Uber or Lyft. Gruel and Stanford (2016) have studied potential traveler behavior when adopting and owning autonomous vehicles. They examined three scenarios: 1. the onset of autonomous vehicles will not change people's behavior; 2. AVs will change people's behavior but not ownership; 3. AVs will change

people's behavior as well as ownership. They found the scenario 3 to be a more sustainable mobility option. In scenario 3, if SAVs are used along with public transportation, they can aid in boosting demand for public transportation by providing the first and last mile connectivity for a traveler and, in turn, counteract sprawl and further encourage MaaS.

The usage of ridesharing services is also dependent on the wait times. When an SAV system is introduced on a subscription basis and is intended to challenge vehicle ownership, the wait times for riders should be within acceptable limits. Therefore, the location of vehicle charging stations is important to minimize wait times of trips. The charging stations act as the home location for the fleet. Vazifeh et al. (2019) used an innovative data-driven approach to obtain the optimal number of charging stations. Using the data consisting of the movements of 1 million users cell phones in the city of Boston, they performed optimization over the network to reduce the minimum distance the driver needs to cover in order to reach the closest charging station. A genetic algorithm was then used to find optimal locations that reduce the number of charging stations by 10% as compared to randomly assigning the locations. Their results were found to be robust to changes with input data and EV penetration rates. Other papers like Ma and Zhang (2018) used the BASS model to predict the number of EVs and the size of charging stations. They solved the objective function, defined as the minimum sum of the cost of waiting time of EV and the cost of operation of charging stations, using 0-1 integer linear programming. The 0 and 1 corresponding to if the site is selected or not.

Some researchers have worked on the dynamic trip allocation of SAVs to further reduce wait times. Hyland and Mahmassani (2018) used 6 optimization strategies to assign

AVs to travelers: 1. Longest idle AV assigned to the first booked traveler; 2. Closest idle AV assigned to first booked traveler; 3. Simultaneous (not sequential) assignment of travelers to idle AVs; 4. Reassigning of AVs from previously assigned traveler to newly unassigned traveler if the latter is nearer to AV; 5. Assigning AV which is about to drop-off the passenger and is nearest to the unassigned traveler; 6. Considering both reassigning of AVs and assigning AV which is en-route drop-off strategies to pick up the nearest unassigned passenger. It is found that the total fleet mileage is reduced from 12.9 miles in the first strategy to 5.4 miles in the last one and the passenger wait times were also decreased considerably. The last 4 sophisticated strategies are particularly effective during peak hours when the utilization of the fleet of AVs is at its peak.

2.4. Chapter Summary

This chapter summarizes some of the relevant work in the area of ridesharing and pricing. Some of the key points are:

- The driver and fuel (gasoline) costs are the most significant costs to ridesharing companies, which are significantly reduced by SAEVs
- The cost of SAV could potentially compete with that of personal vehicle ownership costs
- The general acceptance of SAVs is greatly dependent on the price, and the wait times (availability) of the SAV service

To examine the potential for SAVs to compete with personal vehicle ownership this thesis examines SAVs operated as MaaS in Austin, Texas. To perform this analysis a great deal of data was required, as outlined in the next chapter.

3. DATA

3.1. Overview

For this study, the Capital Area Metropolitan Planning Organization (CAMPO) travel demand model for the Austin and nearby counties was used. Figure 2 below provides an overview of the different counties that are part of the CAMPO model. Each county is further divided into TAZs which makes it convenient to select the region of interest within or between counties for further analysis. The "External" areas represent hypothetical TAZs, which simulate inflow and outflow of traffic from counties outside the model's region of interest. The roadway network is used as links for the traffic flow.

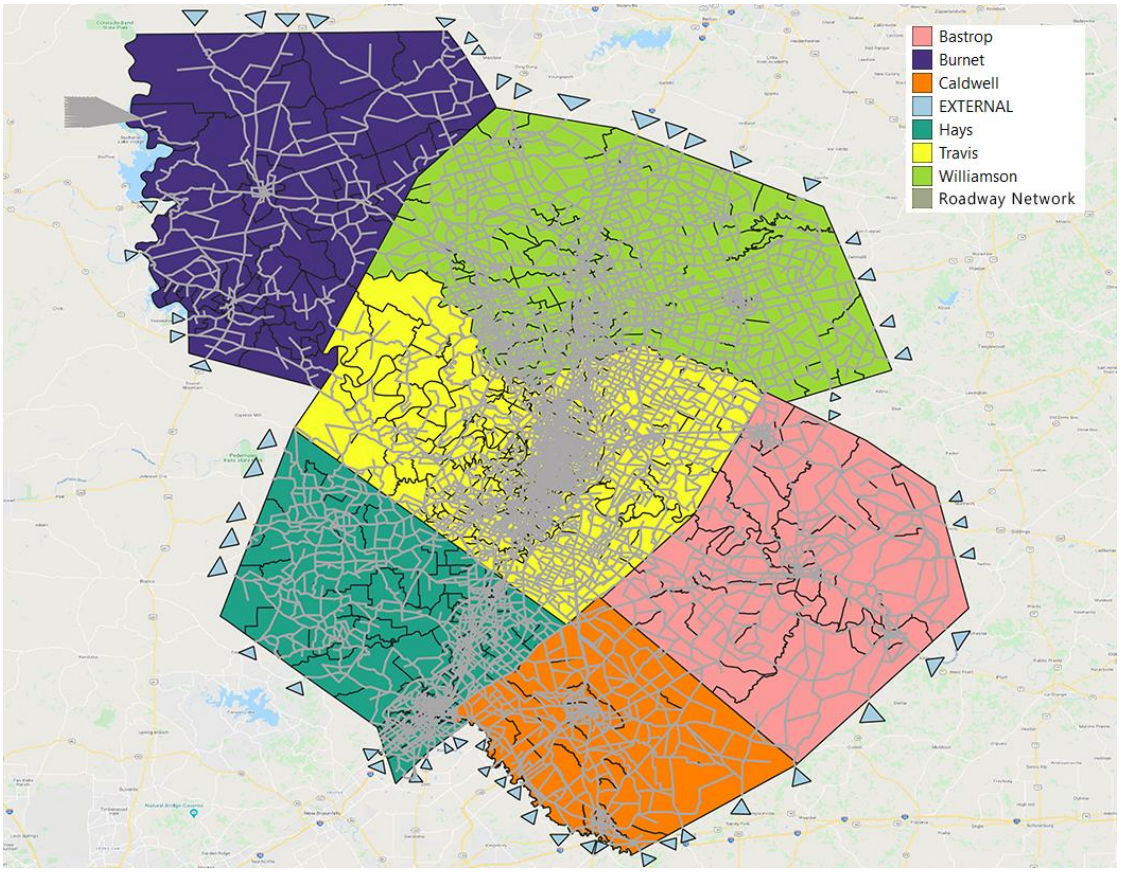


Figure 2 CAMPO Model with Highway Network

3.2. CAMPO Travel Demand Model

The Capital Area Metropolitan Planning Organization (CAMPO) travel demand model for the year 2040 was used to develop the trips between Traffic Analysis Zones (TAZs). Only the projects included in the 2040 regional transportation plan (RTP) were carried into the 2040 roadway network in the CAMPO model. CAMPO developed its population projection for the 2040 scenario with information from the Texas State Data Center (SDC). For the employment growth, the data from the Bureau of Labor Statistics suggests that the economy will continue to produce new jobs and that the employment base of the whole CAMPO region will increase 200 percent to 2.32 million jobs by 2040. CAMPO uses its Demographic Allocation Tool to predict where future population and employment might be located. This tool uses parcel-level data to create an attractiveness rating for each parcel. The data of the transit routes/services included in the 2040 scenario/plan were collected from the service plans of the regional's major transit providers: The Capital Metropolitan Transportation Authority (CapMetro), the Capital Area Rural Transportation System (CARTS) and 38 client-focused transportation providers.

3.3. Data Pre-processing

The CAMPO model consists of 2258 TAZs in total of which only 641 TAZs (see Figure 3) are considered for our case study. These 641 TAZs correspond to Travis County in Austin, Texas, where the population is urban and there are likely to be many early adopters of this type of system. Confining the area to Travis county makes it easy for ridesharing companies to set up charging stations efficiently and evaluate the concept of

MaaS effectively. The SAV service is available for only trips within (both origin and destination) this geographical region (shown in Red in Figure 3).

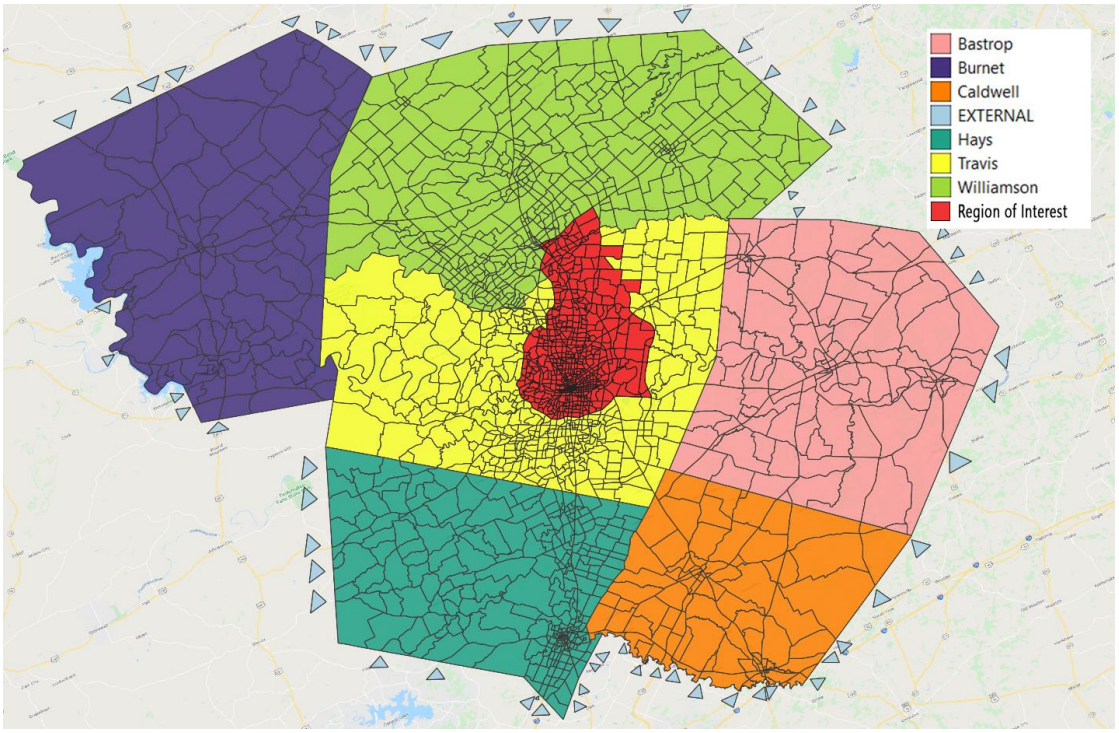


Figure 3 Location of Region of Interest Corresponding to 641 TAZs

A grid system of 3375 (75 x 45) nodes was generated, representing the geographic area used in this study (see Figure 4). The actual origin and destination of trips from the 2040 CAMPO model from the areas outlined in Figure 4 were transposed to the hypothetical grid. The link between adjacent nodes is 0.25 miles in length, making the total area of study 18.5 x 11 miles. This area was divided into 15 zones of 15 x 15 nodes (3.5 x 3.5 miles) and a charging station was placed at the center of each zone. Following the division, the O-D Data from the CAMPO model is filtered to represent TAZs that belong to these 15 zones.

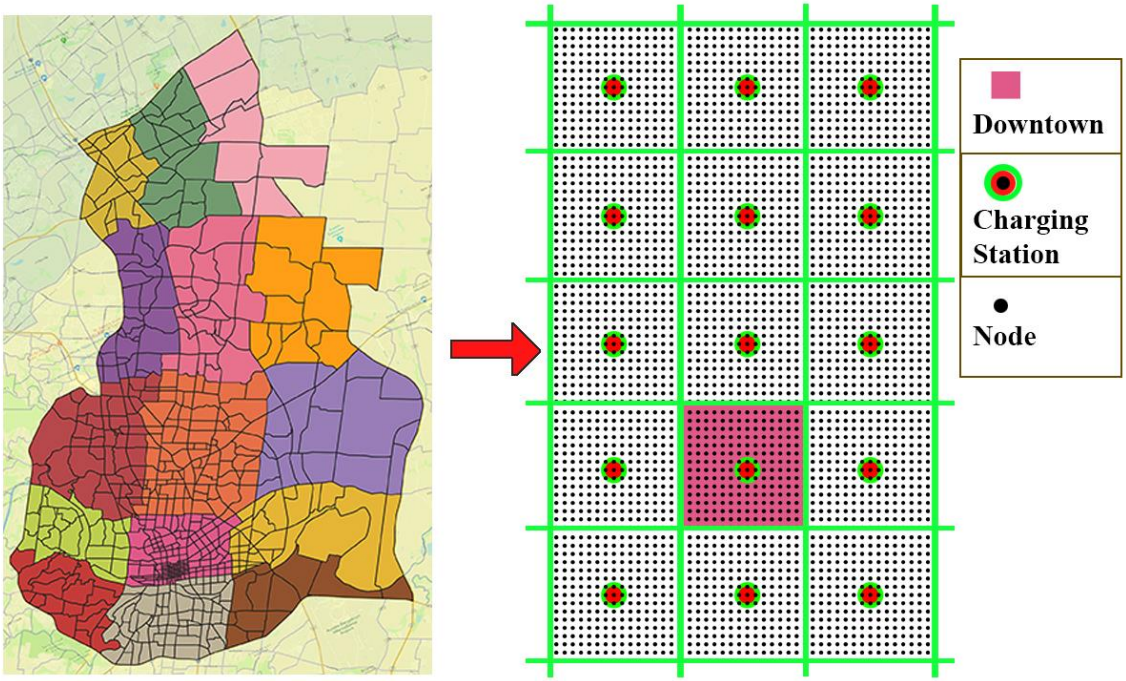


Figure 4 Approximation of Region of Interest onto Grid Network (75 x 45)

3.4. Traffic Speed Characteristics in the Grid Network

Once the grid network was created, the traffic speeds were established using the distance and travel time skim matrices obtained from the CAMPO model. The distance skim consists of the shortest distances along with the highway network from one TAZ to any other TAZ. Similarly, the travel time skim consists of the time it takes to travel from origin TAZ to any other destination TAZ. The travel time skim matrix varies with the time of day, which is divided into 4 time periods: AM peak, Mid-day, PM peak, and Night (shown in Table 1).

Table 1 Division of Day into Four Time Periods

Period	Hours	Duration (min)
AM Peak	06:00 – 09:00	180
Mid-Day	09:00 – 15:30	390
PM Peak	15:30 - 18:30	180
Night	18:30 – 06:00	690

Using the skims, the route level speed is established for all possible TAZ O-D pairs during each time period. The variation of speeds within a day is depicted by histograms in Figure 5. The pairs AM-PM and MD-NT had a similar distribution of speeds. The speeds at NT were found to be higher than the rest of the times of the day. Amongst AM and PM peak, AM speeds were higher. The individual speed characteristics for each time period are presented in Table 2. The average speed across all time periods was close to 30 MPH (0.5 miles per min).

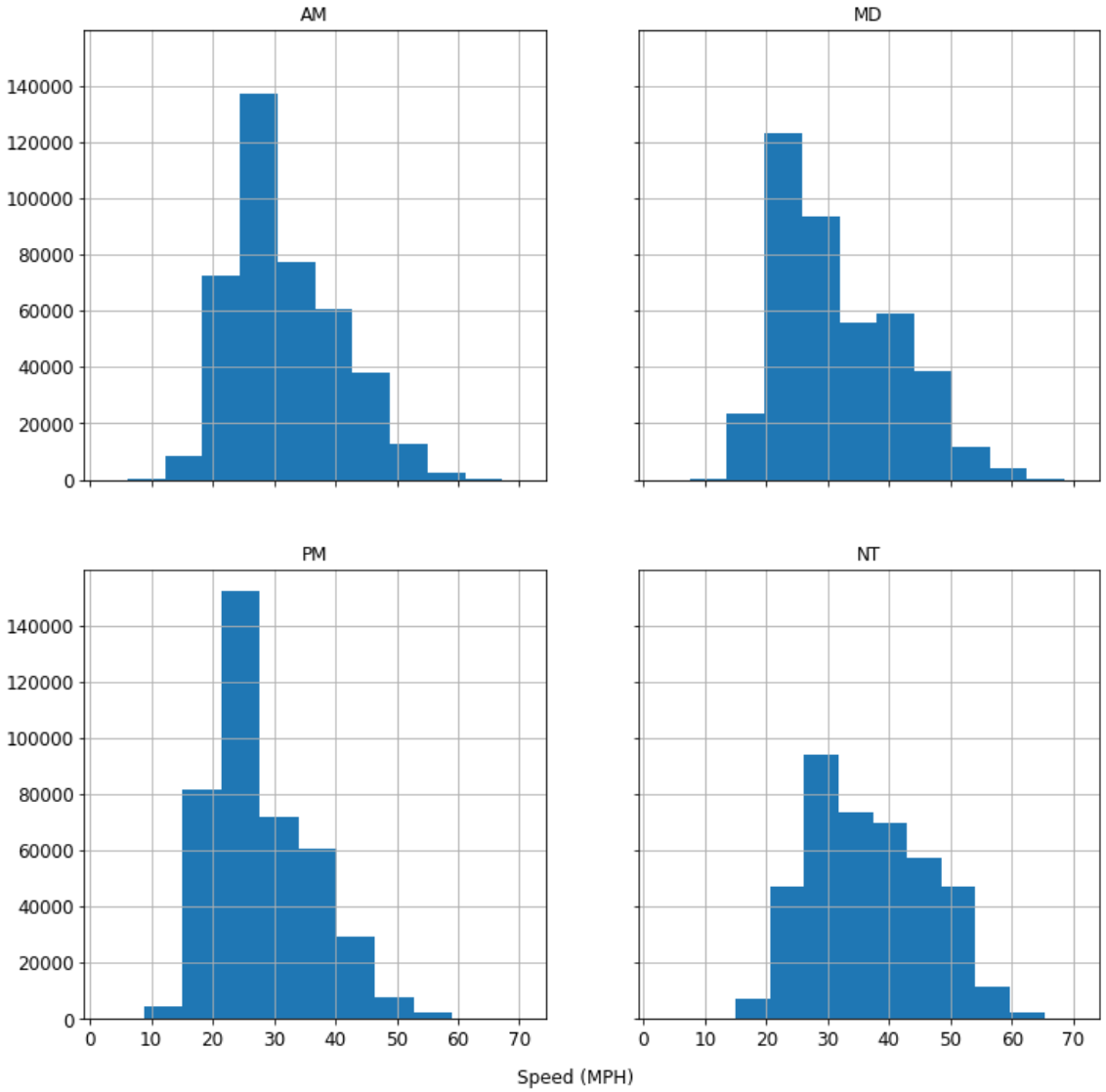


Figure 5 Histograms of Variation of Speeds with Time of the Day

Table 2 Characteristics of Inter-zonal Speeds

Period	Hours	Mean Speed (MPH)	Standard Deviation (MPH)	Minimum Speed (MPH)	Maximum Speed (MPH)
AM Peak	06:00 – 09:00	31.8	8.6	6.1	67.1
Mid-Day	09:00 – 15:30	31.8	9.7	7.5	68.6
PM Peak	15:30 - 18:30	28	8.25	2.5	65.3
Night	18:30 – 06:00	36.9	9.6	15	70

3.5. Chapter Summary

This chapter presents some insights into the Data that is required to conduct simulations. Some key takeaways are:

- The CAMPO Travel Demand model predicts the number of trips in the year 2040 and thereby provides O-D Data
- The study area is restricted to 641 TAZs that belong to Travis County where the population is urban
- A hypothetical grid with 15 zones is overlaid on top of the 641 TAZs
- The average traffic speed in the grid network was close to 30 MPH or 0.5 miles /min

In the next chapters, we use the same hypothetical grid to simulate the ride-sharing environment. The O-D data obtained from the CAMPO model is combined with a suitable

MPR to generate SAV trips every minute. The traffic speed characteristics are used to determine the trip speeds. Combining all this data, the simulation provides the number of SAVs required to satisfy the demand along with charging station capacities. The next chapter will discuss the simulation in greater detail.

4. RESEARCH METHODOLOGY

Simulating ridesharing is a complex task as it involves several interdependent processes such as the generation and assignment of trips, processing the trips (picking up the rider, completing the ride), and the fueling or charging of the vehicles. These processes are time-dependent and stochastic, which makes the ride-sharing system very dynamic and random. To evaluate the feasibility of MaaS with SAVs, the system should also be flexible to changes, since most of the inputs to the system are not constant and are hard to estimate, as it is based on travel patterns. A microscopic simulation model allows for testing of different input variables and evaluation of multiple operational scenarios, thereby helping the TNCs develop better strategies for pricing and operations.

4.1. Simulation Setup

This research developed a simulated environment with flexible and dynamic architecture allowing parameters to be fine-tuned to make the model as realistic as possible. The model also incorporated geographic and physical constraints such as land availability, cost of renting or purchasing the land, and charging station capacity all according to the location in which the system is being deployed. The entire simulation was developed using Python language. Additionally, Python libraries Numpy, and Pandas were used to store, retrieve, and analyze the data.

The simulation has 3 major subparts that update every time step. The time step duration for the simulation is 1 minute. The three subparts are:

1) Trip generation and assignment,

2) Processing of current trips

3) Charging of vehicles.

4.2. Simulation Subtasks

To estimate the number of vehicles required at each charging station inside the city, a mock simulation starting with zero SAVs at every charging station was run for a period of 24 hours (1 day). Each day was divided into 4 periods: AM peak, Mid-day, PM peak, and Night to correspond with the CAMPO model and the travel data from that model. The trips per minute were estimated using the CAMPO model with Market Penetration Rate (MPR) of the ride-sharing system assumed to be 5 percent of total trips generated. The distribution of trips per minute is shown in Table 3. The trips represent all trip purposes and not just home-based work (HBW) trips in the year 2040. Also, all trips had their origin and destination being one in Travis County (the 15 zones in this study). Therefore, the AM Peak had fewer trips per minute than Mid-Day due to the lower overall non-work travel during early morning hours and the many work trips that originated outside of Travis County. The total number of trips in a day for Travis County was close to 2 million. With MPR of 5%, 104,040 trips were generated in 24 hours (equivalent to 72 trips/min). The discussion that follows, and the rest of the thesis, includes only 5% of the trips that were randomly selected for SAV from O-D data.

Table 3 Trips per Minute during Different Time Periods of the Day

Period	Hours	Duration (min)	Trips per minute
AM Peak	06:00 – 09:00	180	58
Mid-Day	09:00 – 15:30	390	120
PM Peak	15:30 - 18:30	180	122
Night	18:30 – 06:00	690	36

For every trip, a trip state is defined to know the status of the trip. In total, 8 trip states are defined, they are:

0 - Trip is assigned

1 - Charging station to the origin

2 - Picking up the passenger

3 - Origin to destination

4 - Dropping off the passenger

5 - Destination to charging station

6 - Docking of SAV

7 - Trip Complete

4.2.1. New Trips Generation and Assignment

The trip origin and destination were based on trips from the CAMPO model. The origin and destination of the trips in the CAMPO model were matched to one of the 15 zones in the simplified model (Figure 4). The exact origin and destination of new trips within each zone were randomized with equal probability of choosing each node. The

generated trips had the same probability of occurring in this model as they do in the CAMPO model for each O-D pair. Each trip was immediately added to a waitlist for the MaaS service. The waitlist ensures that all trips are processed sequentially with priority to trips that were generated early. The dispatch time was defined as the time the trip remains inside the waitlist before it is assigned an SAV.

The maximum total distance that the SAV must cover is the distance from Charge station to Origin, Origin to Destination and Destination to Charging station. Dijkstra's shortest path algorithm was used to calculate the distance between any two nodes. When a new ride request is received, the algorithm checks if any SAVs (with a trip status greater than '4') are available outside charging station and if they are closer to the Origin than the charging station nearest to Origin of the trip. If yes, the SAV is checked if it can complete the trip with its current charge and if assigned the trip if it satisfies the criteria. This process is repeated every time step until the wait time has exceeded 5 min. After 5 min wait time, the next nearest zones were processed. If any SAV outside charging station with a trip status greater than '4' can complete the trip with the current charge, it is assigned immediately. If no SAV is assigned to the trip for 10 min, a new SAV is generated at the charging station nearest to Origin and is assigned to the Trip. This process repeats until all trips in waitlist with wait time greater than 10 min are assigned.

4.2.2. Trips Processing

The trips generated are updated in the next time steps unless the trip has ended with a complete status '7'. In each time step, the location of the vehicle is updated based on the speed of the vehicle, its current location and future desired location. The speed of

each vehicle was multiple of 15MPH (15, 30, 45, and 60 MPH) determined by the time of the day, and the trip origin and destination. The charge of the vehicle was updated accordingly with the amount of distance it travels in each time step. Once the passenger is dropped off, the SAV searches for the charging station nearest to the destination of the trip and proceeds towards it. Meanwhile, the SAV checks for rides nearby and accepts them if it is closer than any other SAV and can safely make the trip and return to the charging station with its current charge.

4.2.3. Charging of Vehicles

Vehicles will proceed to the nearest charging station if their charge is lower than the range of trips or no trips are available nearby. All the SAVs docked inside the charging station (trip state equal to 7) will be charging at a rate of 0.75 miles per minute (45 miles per hour), updated every time step (pluglesspower.com). The maximum charge is assumed to be 250 miles according to Tesla model 3. The SAV leaves the charging station immediately when the range criteria were satisfied, and the trip is assigned to it.

4.3. Day 0 Simulation and Results

By using the methodology in Section 4.2, the Day-0 environment is simulated for 24 hours. Figure 6 shows the flowchart of the Day 0 simulation. Multiple instances of this simulation were run since each simulation gave different results due to the stochasticity of the model. Further, the variation of this model with different speeds was tested.

4.3.1. Variation of SAV Fleet Size with Speed of SAV

The size of the SAV fleet depends partially on the speed of the SAV fleet. For the same number of trips served within a day, the size of the SAV fleet is inversely

proportional to the vehicle speed of the fleet. To understand the variation better simulations with varying speeds (15, 30, 45, and 60 mph) were conducted. Within these simulations, all vehicles were assumed to have a constant speed irrespective of the origin and destination of the trip. At 0^{th} hour, the simulation starts at night with 0 SAVs at all charging stations. After a minimum wait time of 10min, the SAV fleet keeps increasing steadily to match the current demand. This pattern keeps repeating throughout the day and stabilizes at the end of 24 hours. Figure 7 shows a similar pattern for the different speeds simulated. In all the cases, the maximum SAV count was reached before 24 hours of simulation ensuring that simulating the environment for 1 day is sufficient. The marginal reduction of the fleet size decreases with an increase in the speed of the SAV as current running trips become the limiting factor.

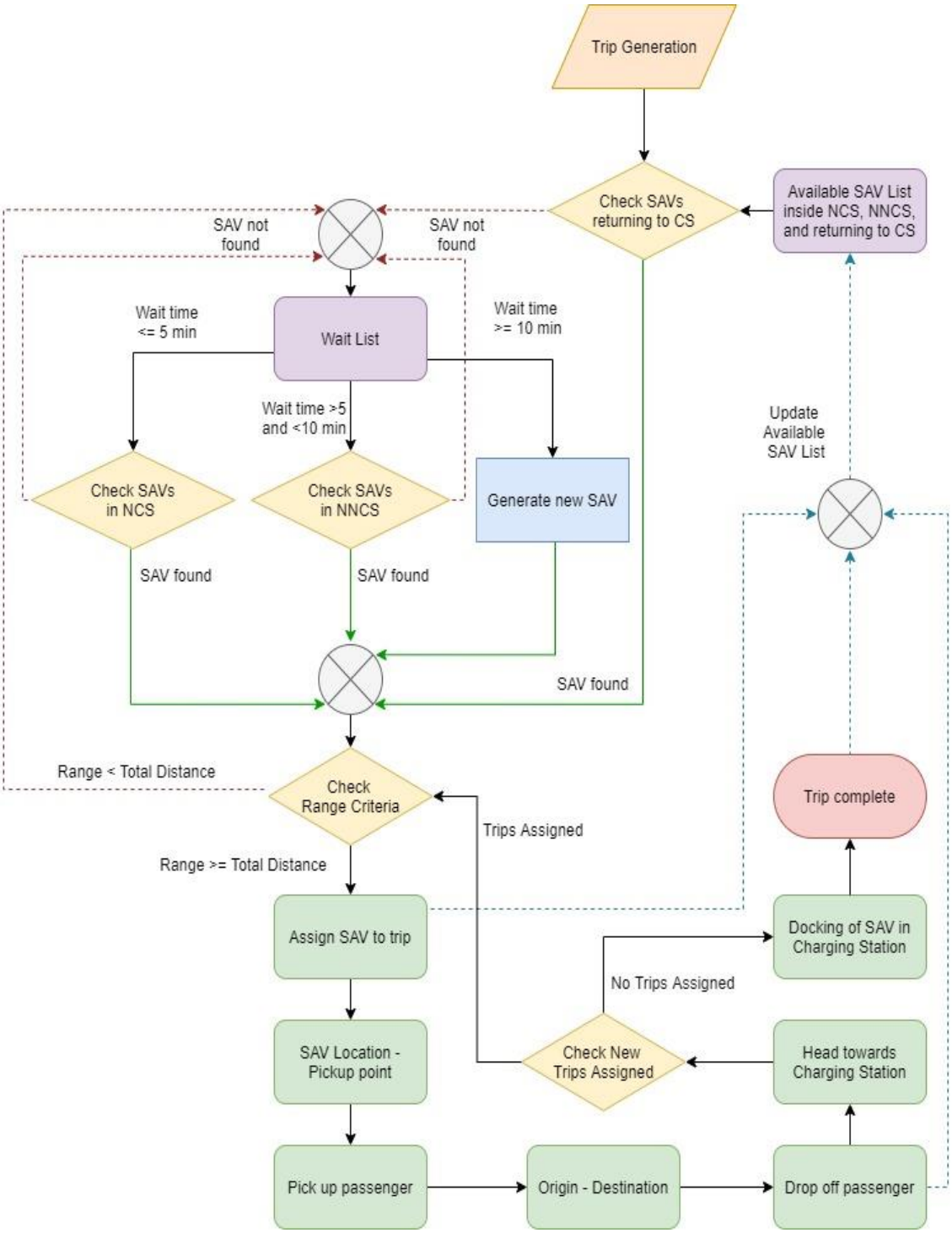


Figure 6 Day 0 Simulation Flowchart

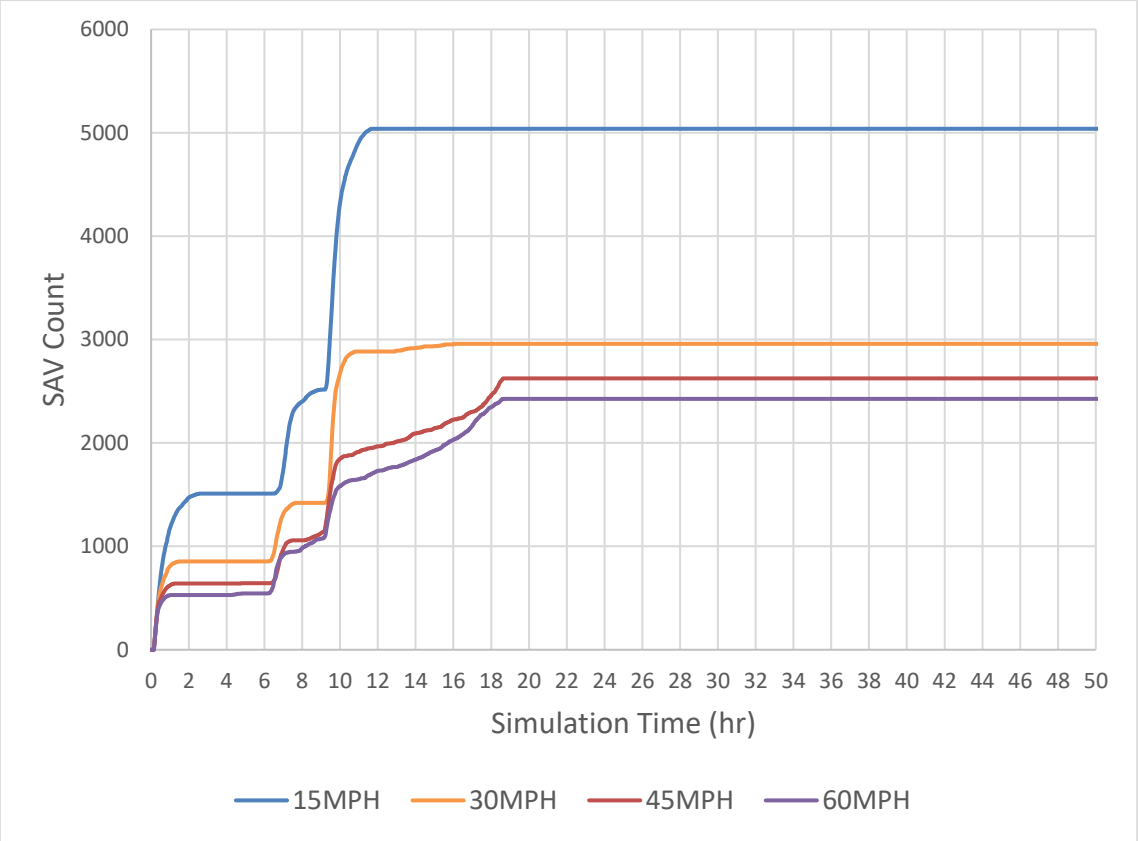


Figure 7 Variation of SAV Fleet Size with Speed

4.3.2. Variation of Current Trips per Minute with Speed of SAV

The number of current running trips is highly correlated with the SAV fleet size. If completing a trip takes no time, then the current running trips graph would be similar to trips generated per minute graph. However, each trip needs some processing (travel) time, during which the SAV is unavailable. This accumulates the trips generated in the previous time steps. In order to service every customer, the fleet size increases. As speed decreases, the time for processing each trip increases thereby increasing fleet size. From Figure 8, it was observed that the variation of current trips with speed was similar to that of Figure 7. The marginal decrease in the current trips diminishes with an increase in speed. At low

speeds, the peak of the curve matches with fleet size, indicating speed as the limiting factor and at higher speeds the peak of the curve is below the fleet size, indicating trip demand as limiting factor.

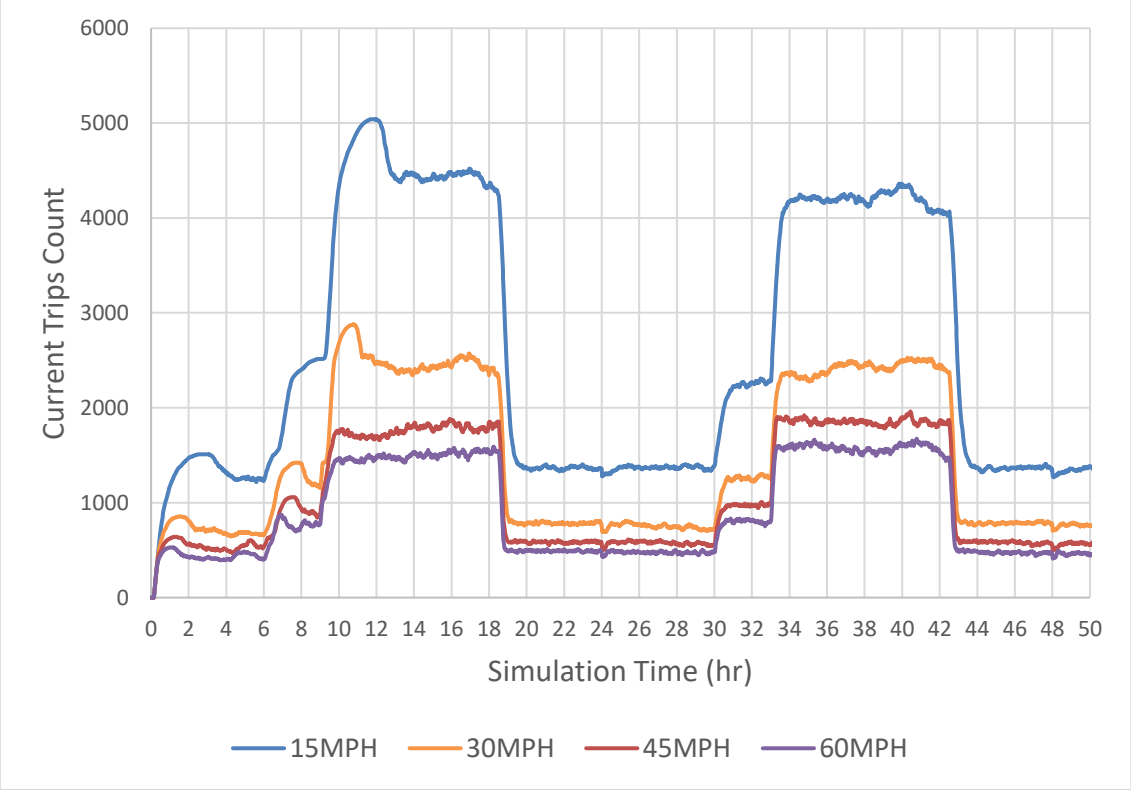


Figure 8 Variation of Current Trips Count with Speed

4.3.3. Time & O-D Dependent Variable Speed Simulation

Having the same speed for all SAVs is not realistic. The speed of vehicles should ideally change with time of the day and O-D. For each time period (AM, MD, PM, and NT), the speeds of the vehicles were determined for all O-D pairs possible. The speeds were calculated using the shortest distance between the O-D and average travel time obtained from the CAMPO model. Their ratio (speed) was approximated to the nearest 15

MPH multiple (15, 30, 45, and 60 MPH). Although few observations had speeds below 15 MPH, 15 MPH was considered the minimum speed due to constraints on simulation size and time. The probability density function (PDF) gives a better comparison of speeds (see Figure 9).

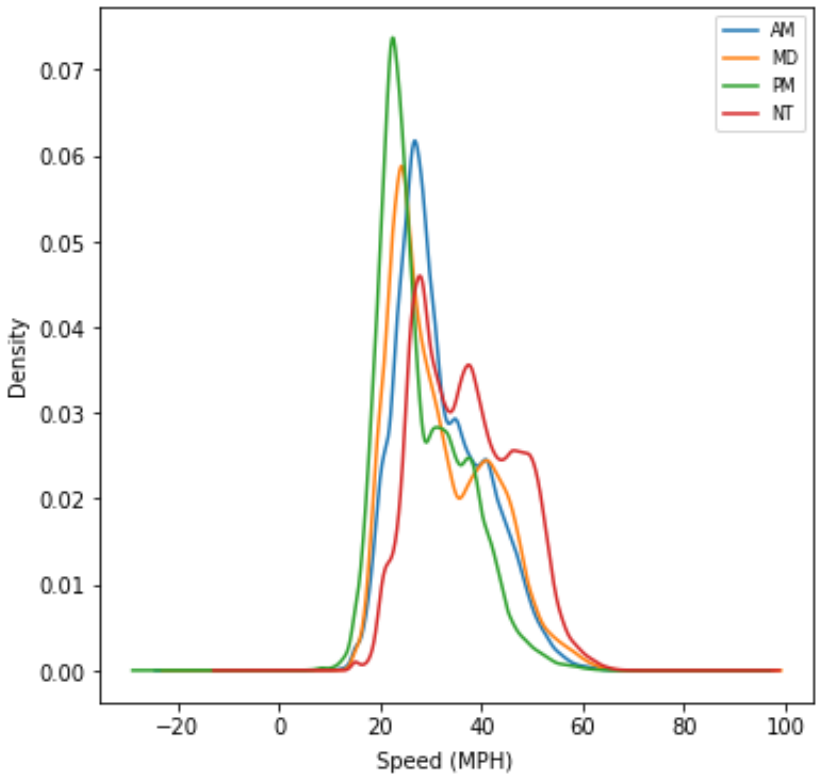


Figure 9 Probability Density Function (PDF) of Speeds with Time of the Day

Like earlier simulations with constant speed, the number of SAVs needed increases initially and then remains constant with time towards the end of the day as shown in Figure 10. For all simulations, a maximum of 3095 vehicles was needed to service every trip in 24 hours, and this was chosen as the number of vehicles to use for the Day-1 simulation. The charging station capacities were adopted based on the simulation results. For every

additional vehicle added, the capacity of the charging stations where the trip originates was increased by 1. The capacities of all charging stations are established by the end of this simulation. The capacities are given as input to the next simulation assuming all SAVs are docked at their respective charging stations with a full charge.

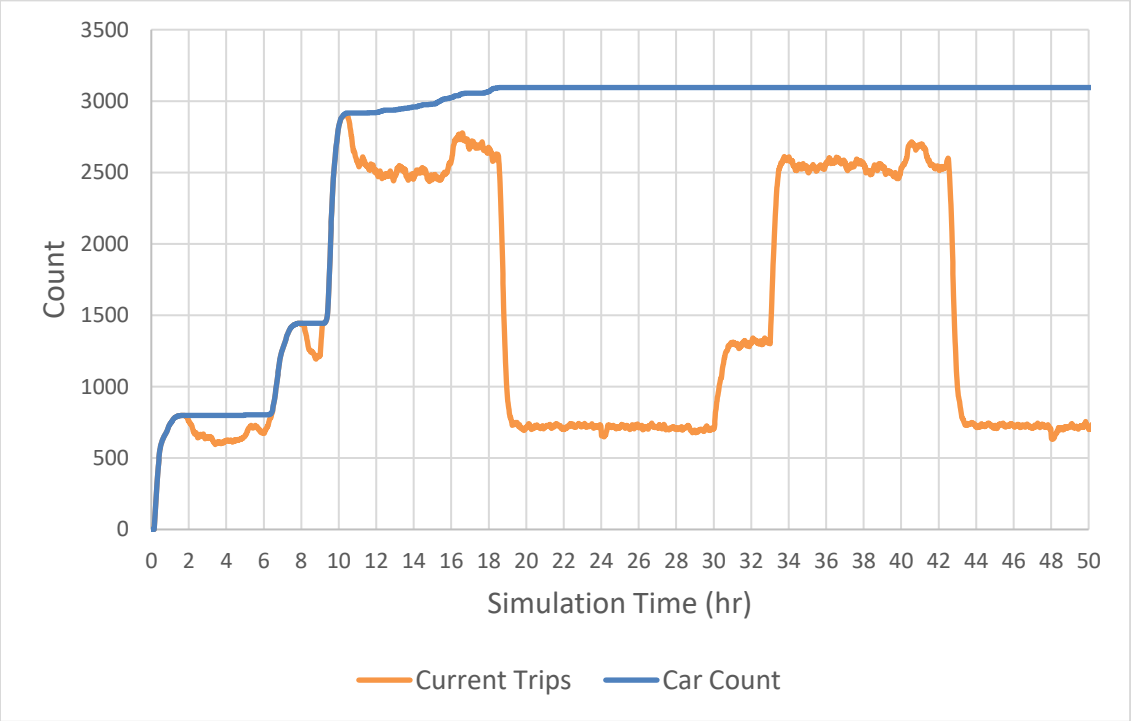


Figure 10 Trip Count vs Car Count in a Day

4.4. Day 1 Simulation and Results

4.4.1. Day 1 Simulation Flowchart

Using the data obtained from Day 0 simulation, the 3095 SAVs were docked according to the charging station capacities and trip needs (as shown in Figure 11). The Day 1 simulation flow chart is shown in Figure 12. The simulation has the same modules of trip generation and assignment, processing, and charging as Day 0 except that the

minimum wait time for dispatch is zero and the maximum wait time for dispatch is assumed to be 8 minutes after which the ride is canceled (by rider). Also, if no SAV is assigned to the trip for 3 minutes of wait time, the next nearest zones to the origin of the trip are checked for available SAV (unlike the 5-minute maximum in the Day 0 simulation).

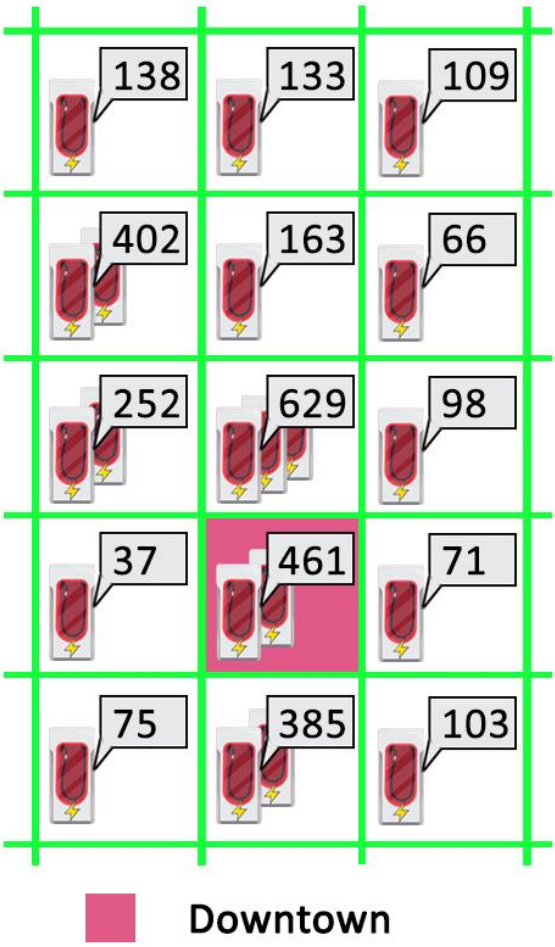


Figure 11 Initial Distribution of SAVs at All Charging Stations

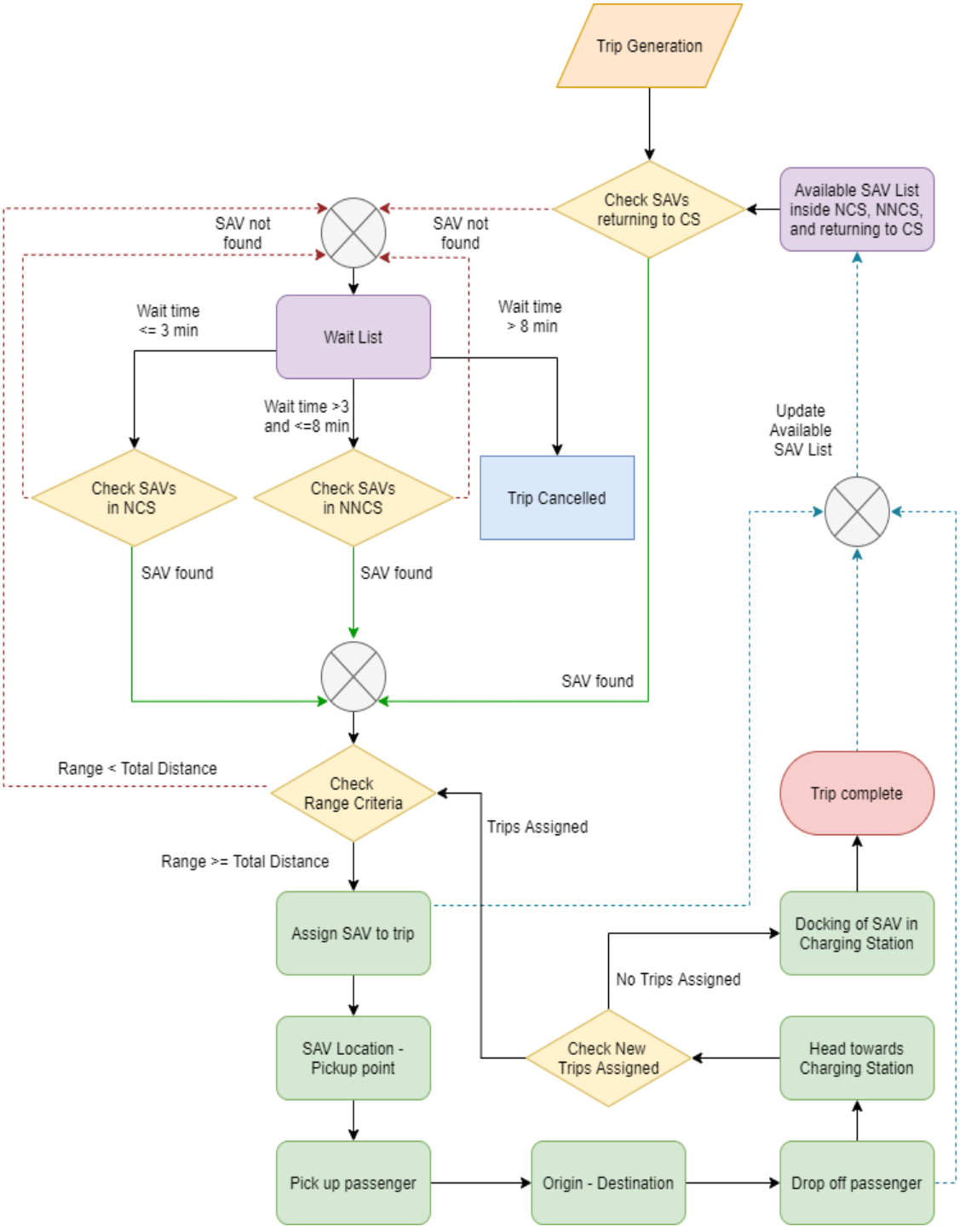


Figure 12 Day 1 Simulation Flowchart

4.4.2. Redistribution of SAVs at Charging Stations

During the Day 0 simulation, the SAVs were generated at charging stations nearest to the origin. Although this method reduces the distance and dispatch times during Day 0 simulation, it is not efficient during Day 1 simulation. The dispatch times and unavailability of SAVs increase if the distribution of SAVs at charging stations is skewed. In some simulation, trips were canceled due to the unavailability of SAVs for more than 8 minutes. To address this issue, during Day-1 simulation, the distribution of SAVs at charging stations was recorded at regular 24hr intervals. The average of these recorded distributions was later used as the initial distribution (see Figure 13).

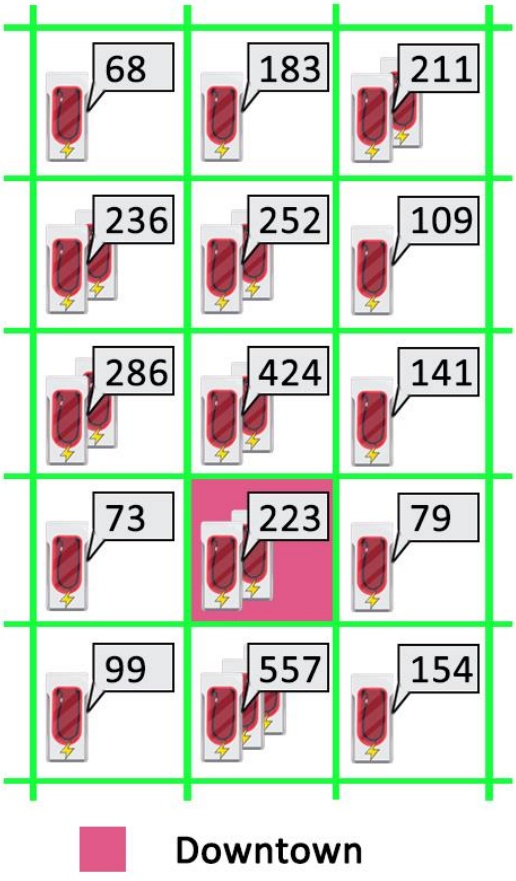


Figure 13 Re-Distribution of SAVs at All Charging Stations

4.4.3. Day 1 Simulation Results

With the newly obtained SAV distribution, the simulation was run for the full day (24 hours). The data of each trip generated and SAV used is stored in every time step into a table in CSV format. On average each SAV makes 34 trips a day, traveling 269 miles in total and 175 miles (5.14 miles per trip) while having a passenger. No trips were canceled during the simulation ensuring that sufficient vehicles were assigned to each charging station. The comparison of trips per min with day 0 simulation is presented in Figure 14. The Day 1 trip per min plot closely retraces the Day 0 simulation except that the peaks are flattened due to the availability of more SAVs initially.

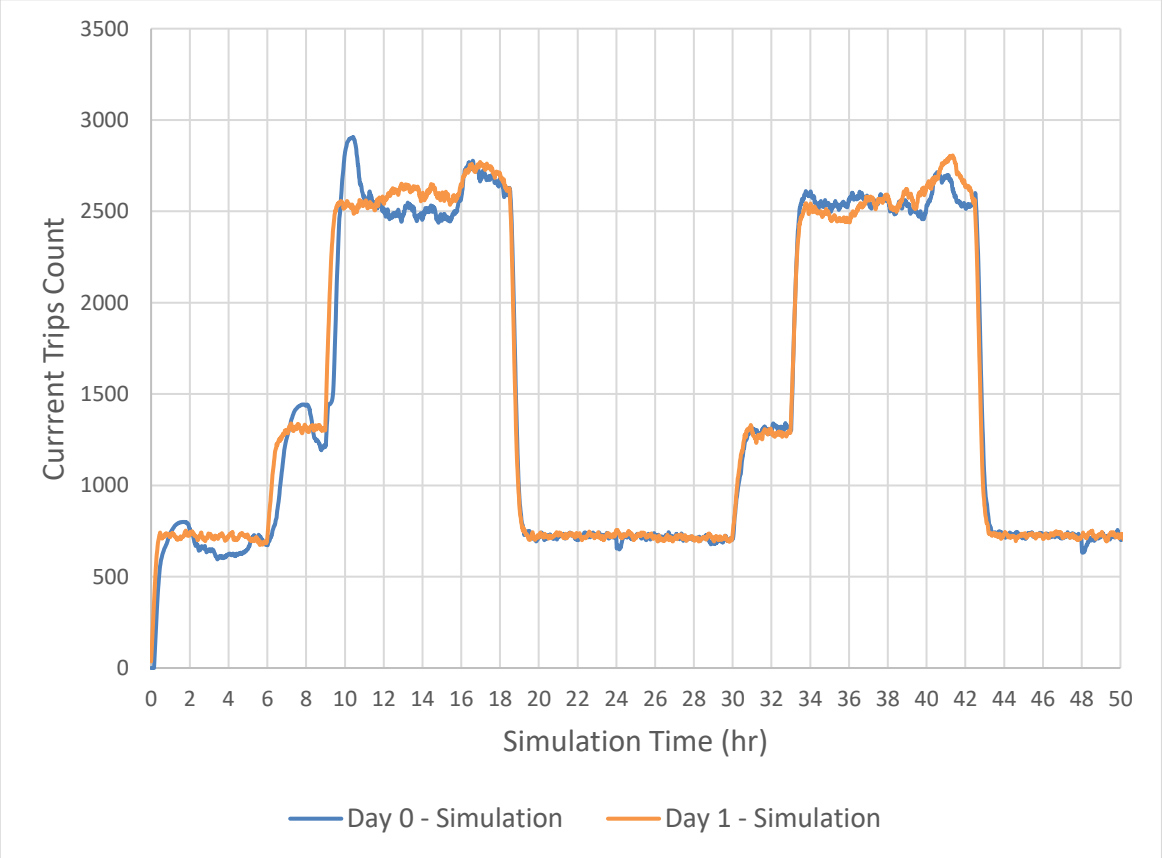


Figure 14 Comparison of Day 0 and Day 1 Simulation Trips per Minute

The wait times for dispatch of SAV during the Day 1 Simulation were recorded. The maximum dispatch time for a trip during Day 1 simulation was 6 min and the overall average dispatch time was 0.213 min (12.8 sec before assigning an SAV to a trip). No trips had dispatch time greater than or equal to 8 min. Therefore, no trips were canceled during the Day 1 simulation. Figure 15 shows the proportions of dispatch times of trips with the percentage of total trips. About 88% of trips had a dispatch time of 0 min i.e. SAVs were assigned immediately to trips. To explore the variation of dispatch times with the time of the day, the average dispatch times for each time period in the day is calculated and is shown in Figure 16. The peaks coincide with the change in the time periods (AM, MD, PM, and NT), indicating a change in O-D distributions.

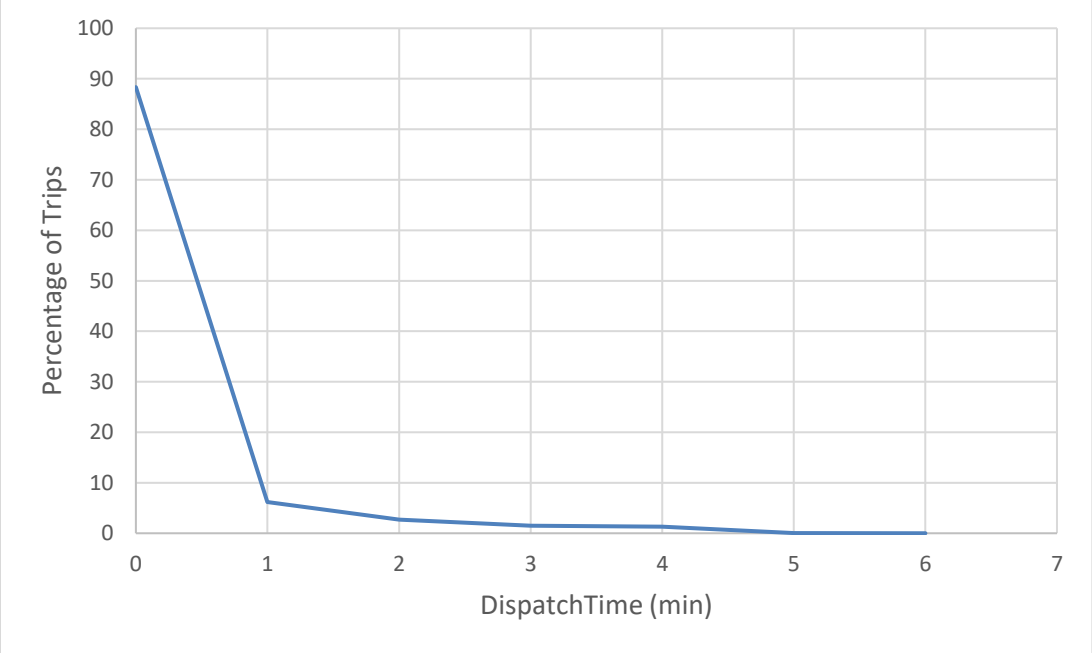


Figure 15 Proportions of Dispatch Times for Trips

The rationale behind the peaks coinciding with the change in time period was due to change of O-D distribution with time period. Therefore, to reduce the peaks in the average dispatch time, the SAVs have to be also re-distributed each day before the start of new time period. The best time for this is during the change from NT to AM when most SAVs are docked at charging stations and charged overnight. However, this would decrease the ration of passenger miles to total miles.

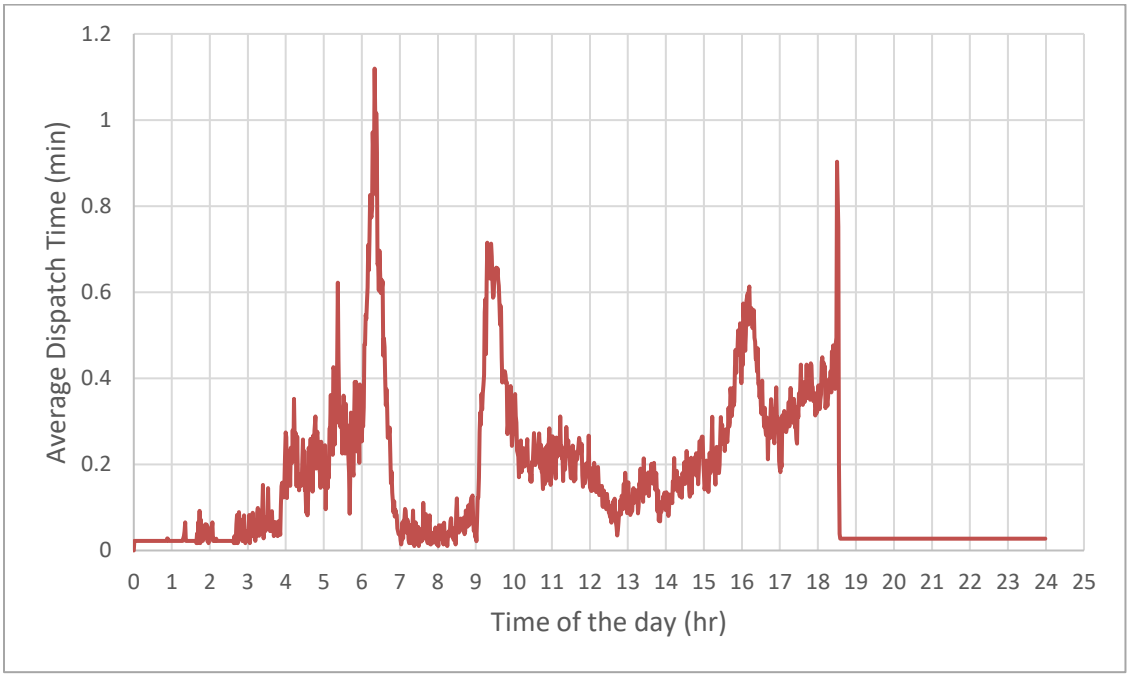


Figure 16 Variation of Average Dispatch Time with Time of the Day

4.5. Chapter Summary

This chapter detailed the simulation model used to stochastically simulate the SAVs. Some noteworthy points are:

- The simulation has three subparts that run simultaneously every time step

(1) Trip Generation, (2) Trip Processing, (3) Charging of SAVs

- The trips were generated according to the CAMPO model and O-D of the trip lies within the hypothetical grid. The average trip length was 4.65 miles
- The Day 0 simulation determines the size of the SAV fleet and provides an initial distribution of SAVs at each charging station.
- The maximum wait time for a trip in Day 0 simulation was 10 min after which a new SAV was generated. A total of 3095 SAVs were generated during the Day 0 simulation
- The Day 1 simulation is used to re-distribute the SAVs amongst the charging stations to reduce wait/dispatch time. With the re-distributed trips, on average each SAV makes 34 trips a day, with 269 total miles and 175 passenger miles

In the next chapter, the fixed and operating costs for SAV service are evaluated and formulated into scenarios to evaluate potential pricing schemes.

5. FINANCIAL ANALYSIS

The financial analysis included the following cost comparisons between SAVs (Tesla) and regular gas-powered vehicles (Ford Focus):

- Fixed Costs
    - Vehicle purchase
    - Charger
    - Land
    - Ownership costs (Insurance + Registration + Taxes & Fees)
    - General administration
    - Attendants
- Operating Costs
    - Fuel
    - Maintenance, repair, and tires
    - Cleaning

5.1. Fixed and Operating Costs

The simulations established the fleet size of SAVs needed to serve the given demand (based on MPR), and the usage statistics of each SAV. Next, several assumptions were necessary for financial analysis. The fixed and operational costs for the SAV were calculated first to see if SAVs could compete with existing TNCs. For SAV, a Tesla Model 3 is taken as the reference vehicle and for the personal vehicle, a gasoline Ford Focus and electric Tesla Model 3 were taken as the reference vehicles.

The SAV fleet costs were then compared with personal vehicles to evaluate the financial viability of SAV as MaaS. For the comparison with vehicle ownership, the traveler was assumed to own a gas-powered Ford Focus vehicle. The cost per SAV per traveler and vehicle ownership costs per individual are presented in Tables 6,7,8 and are further elaborated in detail.

The vehicle life for an SAV was based on its daily usage. On average, each SAV traveled 269 miles a day, which equals 98,185 total miles per year per vehicle (assuming similar usage on weekdays and weekends). The Tesla Model 3, which is used as the SAV for calculations, can run for 250,000 miles in its entire life span, although the company's CEO Elon Musk claimed that the vehicle can run for 1 million miles (electrek.co). Based on the 98,185 miles per year, this means approximately 2.5 years (2.55 years) life before retiring the vehicle. The vehicle life for both the personal vehicles Tesla Model 3 and Ford Focus was assumed to be 5 years with 15,000 miles per year as usage. On average people retain the same vehicle for 5-7 years which is similar to the average auto loan period (autolist.com).

The vehicle purchase costs for Tesla model 3 and Ford Focus were obtained from their respective websites (tesla.com, and ford.com). The Tesla Model 3 comes equipped with a variety of sensors. These include eight optical cameras, twelve ultrasonic (sonar) sensors, and a radar. The sensor information combined with the robust vehicle control algorithms makes Autopilot possible. The company claims that the vehicle has all the hardware essential for self-drive capability, which is anticipated to be allowed in the future

(www.tesla.com/autopilot). Therefore, the cost of the fully autonomous Tesla model 3 was assumed to be the same as the standard Tesla Model 3.

Since the Tesla SAVs are electric, they need charging stations with chargers. The cost of the Tesla wall connector charger is $500. Each SAV needs a space for charging itself, the fixed and operational costs of charging stations were evaluated. The fixed costs included the cost of purchasing land for charging stations based on each charging station's capacities established through Day 1 simulation. As per Loeb and Kockelman (2019), the land acquisition costs varied with the location of charging stations. The land costs were divided into three categories High, Mid, and Low based on proximity to downtown Austin and the categorization is presented in Figure 17. The average land acquisition cost per SAV was then evaluated according to Table 4. Therefore, the land acquisition cost was assumed to be $3201 per vehicle space in Travis County, Austin.

Table 4 Calculation of Average Land Acquisition Cost

Category of Site	**Acquisition Cost per vehicle space**	**SAV Count**	**Cumulative Cost**
Low	1980	1059	2096820
Mid	3460	1813	6272980
High	6900	223	1538700
Total		3095	9908500
Average Land Acquisition Cost			**3201**

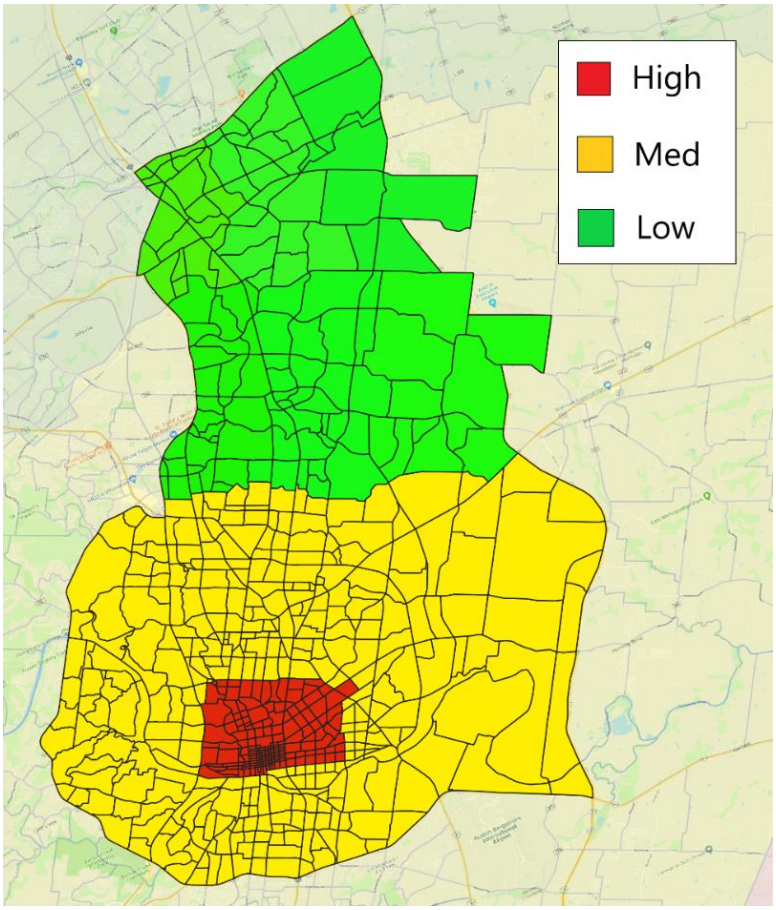


Figure 17 Variation of Land Acquisition Costs

For every charging station, the variable costs included administrators and attendants to maintain the charging stations and service vehicles. 1 General Administrator (GA) and 2 attendants (AT) were appointed per charging station. The wages were assumed to be $25/hour for GA and $12/hour for the AT. For the 15 charging stations and 3095 vehicles, this results in a cost of $2.91 and $2.79 per vehicle per day for GA and AT respectively. This amount, when multiplied with 365 days, gives $1061 and $1019 per year. For 2.55 years (AV life), this totals a cost of $5297 per vehicle. The calculations are shown in Table 5.

Table 5 Calculation of Charging Station Maintenance Cost

Employee	Number	Cost per hour	Cost per day	No of charging stations	Cost per day per vehicle	Cost per vehicle per year	Cost per vehicle (2.55 years)
GA	1	25	600	15	2.91	1061.39	2703
GT	2	12	576	15	2.79	1018.93	2594
Total					**5.70**	**2080**	**5297**

The fuel/electricity costs, maintenance costs, and ownership costs for the electric vehicles were taken from "Your driving costs 2018" report released by AAA (American Automobile Association) for SAV and from the Edmunds website (edmunds.com) for the Ford Focus. As per the AAA report, the electricity costs for electric vehicles were calculated at a rate of $0.125 per kWh, and the maintenance, repair, and tire costs are $0.076 per mile. These costs were for conventional electric vehicles (not AVs), but Wadud (2017) states that the net changes in these costs for AVs are small and can be neglected. Cleaning costs of vehicles were assumed to equal 8 cents per mile (Litman, 2019). The cleaning costs are additional to maintenance repair costs as each SAV is used by a group of riders, therefore cleaning the SAV frequently would be necessary.

Table 6 Vehicle Costs for Tesla SAV

Costs	**Tesla (SAV)**	
	Source	**Cost per vehicle**
Fixed Costs		
Vehicle purchase	Tesla website (tesla.com)	39000
Charger	Tesla website (tesla.com)	500
Land	Loeb and Kockelman, 2019 (Table 4)	3201
Ownership costs (Insurance + Registration + Taxes & Fees)	AAA (2018)	3109
General administration ($25/hr)	Table 5	2703
Attendants ($12/hr)	Table 5	2594
Fixed costs for vehicle life (2.55 years)	**51868**	
Fixed costs per year	**20340**	
Fixed costs per year per traveler	**2034**	
Fixed costs per month per traveler	**169.50**	
Operating costs per mile		
Items	**Source**	**Cost per vehicle per mile**
Fuel (Electricity)	AAA (2018)	0.04
Maintenance, repair, and tires	AAA (2018)	0.08
Cleaning	Litman, 2019	0.08
Operating costs per mile	**0.20**	

Table 7 Vehicle Costs for Personal Tesla

Costs	**Tesla (Personal)**	
	Source	**Cost per vehicle**
Fixed Costs		
Vehicle purchase	Tesla website (tesla.com)	39000
Charger	Tesla website (tesla.com)	500
Ownership costs (Insurance + Registration + Taxes & Fees)	AAA (2018)	6105
Fixed costs for vehicle life (5 years)	**45605**	
Fixed costs per year	**9121**	
Fixed costs per year per traveler	**9121**	
Fixed costs per month per traveler	**760.08**	
Operating costs per mile		
Items	**Source**	**Cost per vehicle per mile**
Fuel (Electricity)	AAA (2018)	0.04
Maintenance, repair, and tires	AAA (2018)	0.08
Operating costs per mile	**0.12**	

Table 8 Vehicle Costs for Personal Ford Focus

Costs	**Ford Focus (Personal)**	
	Source	**Cost per vehicle**
Fixed Costs		
Vehicle purchase	Ford website (ford.com)	18800
Ownership costs (Insurance + Registration + Taxes & Fees)	Edmunds website (edmunds.com)	8905
Fixed costs for vehicle life (5 years)	**27682**	
Fixed costs per year	**5536**	
Fixed costs per year per traveler	**5536**	
Fixed costs per month per traveler	**461.33**	
Operating costs per mile		
Items	**Source**	**Cost per vehicle per mile**
Fuel (Gasoline)	AAA (2018)	0.08
Maintenance, repair, and tires	AAA (2018)	0.08
Operating costs per mile	**0.16**	

Since each AV completes approximately 34 trips per day and a person travels on average 3.37 trips per day (NHTS 2017), an average of 10 travelers use a single SAV per day. This amounts to a fixed cost of $169.2 per month and an operating cost of $0.20 per mile for each traveler. This forms the base case where the business operator charges this price to customers and there is no profit for operating an SAV fleet. The total fixed and operating costs for the Personal Tesla are $760.08 per month and $0.12 per mile, and for Personal Ford Focus are $461.36 per month and $0.16 per mile.

5.2. Subscription Charges

In the article "Uber and Lyft return to Austin: What's changed, and why it's important", it was noted that Uber and Lyft charged $1 per mile and 20 cents per minute. (curbed.com). In the model, the vehicles are moving at an average speed of 0.5 miles per minute (30 mph). So, traveling 1 mile takes 2 minutes making the total fare $1.40 per mile. This price is further used to compare SAVs with other TNCs in additional scenarios. The first scenario examines the prices TNCs would charge to break even. This equates to a subscription charge of $169.5 per month and $0.20 per mile the SAV travels. However, only 65% of the SAV miles were passenger miles, which makes the charge per mile close to $0.31. The next two scenarios investigate alternate subscription-based pricing schemes.

Each AV travels 98185 miles per year of which 63820 are passenger miles per year, which when divided by 12 months and 10 shared users is equivalent to 532 miles traveled per rider per month. As per NHTS 2017, a typical traveler covers an average of 38.98 miles per day or 1169 miles per month. The next two scenarios keep the monthly subscription costs the same and calculate the maximum per mile costs that can be charged to compete with the cost of vehicle ownership while providing profits to the SAV subscription company. Scenario 2 considers the average miles traveled by each traveler to be 532 miles, estimated through simulation. Scenario 3 considers the average miles traveled by a traveler to be 1169 miles according to NHTS. The subscription costs per month, charge per mile, and profits per month per traveler for all three scenarios are presented in Table 9.

Table 9 Profits with SAV Subscription

Cost	Scenario 1 (Base scenario)	Scenario 2 (532 miles)	Scenario 3 (1169 miles)
Subscription per month ($/month)	169.5	169.5	169.5
Charge per mile ($/mile)	0.31	0.71	0.41
Profit per month per traveler ($/month/traveler)	0	213	117

5.3. Comparison of Alternatives

The three scenarios represented by the charge per mile ($0.31, $0.71, and $0.41) are plotted against the vehicle ownership costs (personal Ford Focus and Tesla Model 3) and the current price charged by a TNC ($1.4/mile) as shown in Figure 18 below. Using the graph, a traveler can decide the most economical option based on their monthly usage. For example, in scenario 2, if a rider travels less than 246 miles in a month, the current price charged by TNC would prove to be cheaper. However, if the rider decides to travel more than 246 miles and less than 532 miles in a month, buying monthly subscription by AVs would be the cheapest option. If they drove more than 532 miles per month, the cheapest option would be owning a personal vehicle (Ford Focus). Similarly, in scenario 3, below 171 miles per month the TNC, between 171 and 1169 miles per month the SAV subscription and above 1169 miles per month vehicle ownership (Ford Focus) are the most economical options.

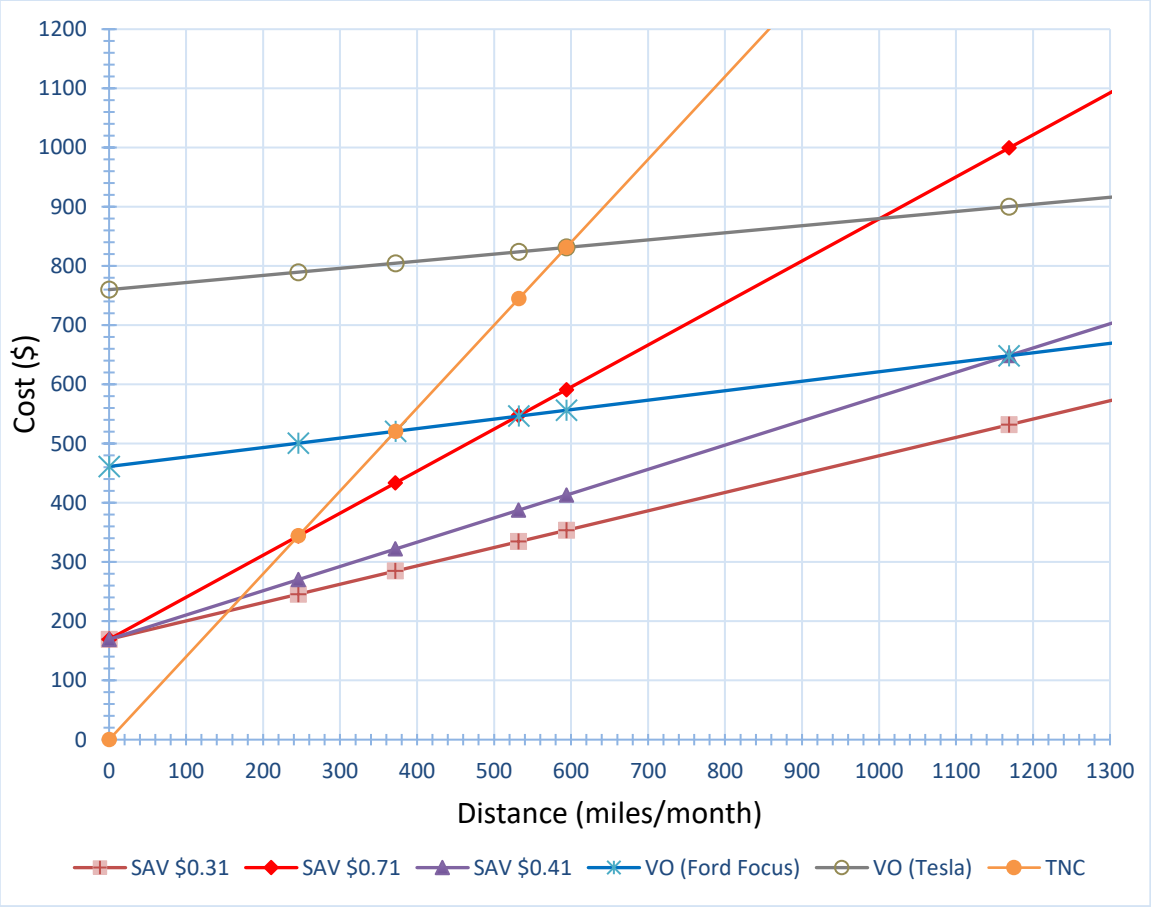


Figure 18 Comparison of Costs in 3 Scenarios

5.4. Chapter Summary

In this chapter, various costs were calculated to evaluate the fixed and operating costs of SAV, Personal Ford Focus, and Personal Tesla. Three scenarios were formulated to determine subscription pricing. Some important takeaways from this chapter are:

- SAVs distribute the fixed costs of riders by sharing it amongst a group of riders thereby reducing the financial burden for each rider. The fixed cost for SAV subscription was close to $170 per month, whereas, for Personal Ford Focus and Personal Tesla, it was observed to be $460 and $740 per month, respectively
- Scenarios 1, 2 and 3 were formulated and their operating costs were found to be $0.31, $0.71, and $0.41 per mile respectively
- The profits per month per traveler were found to be $213 in Scenario 2, and $117 in Scenario 3
- The comparison of costs helps users make a rational choice and choose the best alternative according to monthly requirement/usage.

Thus, the financial analysis shows that the subscription based SAV ridesharing service is financially viable. At the same time, it is also economically beneficial for a vast segment of riders.

6. CONCLUSION AND FUTURE RESEARCH

AVs can truly transform the ridesharing environment by providing more accessibility and improving safety to riders. The concept of SAV where a single AV is shared amongst multiple users has many additional benefits. Firstly, SAVs provide the comfort of car travel while lowering the cost to riders. SAVs improve the reliability of the ridesharing service as they offer a steady supply at any time of the day. Electric SAVs have fewer emissions and further cost benefits than gas-powered SAVs. Therefore, various companies are experimenting in the field of SAV including Google, Tesla, and GM. A subscription-based pricing scheme instead of pay-per-ride pricing would ensure an unswerving demand for SAV companies. Additionally, they can help rider's transition from personally owned vehicles to transportation services on demand by switching to a single ridesharing service.

The current research examined how SAVs might operate in a city with subscription pricing. The research stands unique as it explores the concepts of subscription pricing for SAVs and performs financial analysis from the perspective of both business operators and travelers. The research also presents a proof of concept by simulating ridesharing and evaluating pricing in the city of Austin, Texas. The simulation model is flexible to changes and can be used to analyze different scenarios/cities by varying its parameters such as grid size, node distance, speed, etc. By changing these parameters, the model can be used to evaluate the subscription pricing, and viability of SAV operations for any other city/town based on O-D data, speed characteristics, and MPR.

For the city of Austin, Texas, the simulations showed that 3100 SAVs could serve 5% (MPR) of total all-purpose trips with an average dispatch time of 13 seconds and maximum dispatch time of 6 min. During the simulation, a single SAV on average traveled 532 miles making 34 trips per day, substituting 10 conventional vehicles operating in an urban area. This is consistent with the results shown by Fagnant et. al (2015) whose study claimed SAV could replace 9 conventional vehicles. The financial analysis was presented in the form of three scenarios that compared the TNC, SAV, and vehicle ownership alternatives.

The three scenarios showed that as the distance traveled per rider per month increases, profits to SAV company increases, and savings compared to ownership costs reduce. More specifically, scenario 1 determined the least per mile operating price to ensure no loss for SAV company, which was equal to $0.31 per mile. With $0.31 charged as operating cost, the SAV subscription would be the best alternative for riders traveling more than 155 miles per month. Scenario 2 was used to decide the per-mile operating costs if the SAV companies want to ensure people maintain the way they travel currently (i.e. according to simulation). The results show a per-mile cost of $0.71 and the SAV subscription would be suitable for riders traveling more than 246 and less than 532 miles per month. Scenario 3 determined the maximum per mile operating costs if SAV companies want to compete with vehicle ownership. It was observed to be $0.41 per mile, which is close to time-based per-mile operating cost ($0.40) currently charged by TNCs. Consequently, the SAV subscription would be the most economical alternative for riders traveling more than 171 and less than 1169 miles per month.

The difference in operating costs of scenarios 2 and 3 may be reduced by optimization of the service, reduction of costs, and increase of profits. The improvements in operations include better techniques for assigning SAVs to trips, efficient distribution of vehicles in charging stations, and re-allocation of empty SAVs. These techniques either improve the ratio of passenger miles to total miles or reduce the wait times. The reduction of costs could also be brought about by having more SAVs at charging stations with lower costs and making collaborations with vehicle companies to reduce vehicle and charger costs (e.g. Uber collaboration with Volvo). The cost of electric vehicles is on a decline due to improvements in vehicle and battery technology. Also, the cost of sensors required for AV is on a decline and is expected to decrease further following Moore's Law. Besides, the profits for SAV companies can be increased by incorporating surge pricing models that have higher costs during peak times. This also helps to keep the demand in check during peak hours. Also, tier-based pricing schemes can be introduced with additional benefits to top tier people at some minimal cost. Therefore, the future of SAV service seems promising economically to both users and companies.

Apart from the direct cost benefits, SAVs also provide benefits in the form of the value of time savings. The rider can make the best use of their time whilst making the trip as they are not engaged in the driving. Considering economic/financial benefits, the SAV service can be expected to be a strong competitor to vehicle ownership and encourage MaaS. However, the comfort and reliability that a personal vehicle provides are hard to achieve. A system where travelers express an interest in a trip in the future could improve the reliability of the service as the SAV fleet could re-organize efficiently according to the

schedule of rides. A shift from trip-based SAV service to time-based service, where SAV could be rented on time/day basis would improve the reliability and comfort to the travelers. A free market is essential just like any other good or service so that travelers have other modes of transport if they do not like to use SAV service or the SAV service is unavailable. Perhaps multiple SAV companies could be a solution. This would force companies to keep their prices competitively, limit profit margins thereby challenging vehicle ownership and genuinely promoting MaaS.

The current research had to make many assumptions and does not consider various extra costs that may be incurred, such as installing cameras to monitor the behavior of passengers (Broussard, 2018), or installing wireless communication between vehicles (Litman, 2019). Since the area of study is restricted to Travis County, only trips that originate and end within this area are considered resulting in fewer long-range trips. Extending the area of study to larger areas may provide additional information in handling long-range trips that may require a different arrangement of SAVs.

Some of the extensions to this research could include examining surge pricing models to reduce the demand during peak hours thereby reducing fleet size and increasing profits. Since all vehicles are assumed to be electric, they must return to charging stations intermittently. This increases the number of empty vehicle miles traveled. The optimization of the location of charging stations as well as efficiently allocating vehicles to pick-up and drop-off passengers can reduce empty vehicle miles traveled thus increasing profitability. The concept of carpooling opens several other dimensions to the

research such as optimal transfers, operating high occupancy vehicles, and efficient demand management.

This research examined the case where the ridesharing company owns and operates all of the SAVs/vehicles. If the diffusion of autonomous vehicles in the public is considerable, the individual owners could share their vehicle with a ridesharing company, when it is not in use. This could, in turn, increase the demand and diffusion of AVs as they generate additional revenue when the vehicle is not in use by its owner, making SAVs more common and acceptable. Although the future remains unpredictable, the benefits especially economic that SAV could provide are definite. With greater diffusion of AVs, the possibilities with SAV services in all walks of transportation are enormous. Developing algorithms for such services and evaluating pricing would indeed be a challenging and interesting field of research.

REFERENCES

1) HTF Market Intelligence (2019), "Global self-driving car market (2018-2023)", https://www.htfmarketreport.com/reports/1304659-global-self-driving-car-market-2. Last accessed May 02, 2019.

2) National Highway Traffic Safety Administration (2015), 2015 Summary of Motor Vehicle Crashes (Final Edition) Traffic Safety Fact Sheet, https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812376#:~:text=In%202015%2C%20there%20were%20an,estimated%202%2C443%2C000%20people%20were%20injured.&text=An%20average%20of%2096%20people,one%20fatality%20every%2015%20minutes. Last accessed June 10, 2020.

3) Statista (2019), "Ride-Hailing", https://www.statista.com/outlook/368/109/ride-hailing/united-states. Last accessed May 02, 2019.

4) Ulama, D. (2016), "Taxi & Limousine Services Market Research Report," IBISWorld Industry Report.

5) Waymo (2019), https://waymo.com. Last accessed May 02, 2019.

6) Schoettle, B. and Sivak, M. (2015) Potential Impact of Self-Driving Vehicles on Household Vehicle Demand and Usage; Report No. UMTRI-2015-3; University of Michigan Transportation Research Institute: Ann Arbor, MI, USA.

7) Fagant, D., Kockelman, K. and Bansal, P (2015). Operations of Shared Autonomous Vehicle Fleet or Austin, Texas, Market. Transp. Res. Rec. 2015, 2536, 98–106.

8) Isaac G. Freedman et. al (2018) “Autonomous vehicles are cost-effective when used as taxis”, Injury Epidemiology, 5-24.

9) Jana Lynott (2018), “Universal Mobility as a service: A bold vision for harnessing the opportunity of disruption”, AARP Public Policy Institute.

10) Juniper and moovel (2017), “Exploring Mobility-as-a-Service (MaaS): The New Era of Urban Mobility”, https://nabsa.net/wp-content/uploads/2017/09/Exploring_Mobility_as_a_Service.pdf. Last Accessed June 10, 2020.

11) Nykvist, Björn, Nilsson, Mans (2015). “Rapidly falling costs of battery packs for electric vehicles”. Nature Climate Change 5, 329–332.

12) Benjamin Loeb, Kara M. Kockelman (2019), “Fleet Performance and cost evaluation of a shared autonomous electric vehicle (SAEV) fleet: A case study for Austin, Texas”, Transportation Research Part A 121, 374-385.

13) Aditi Moorthy, Robb De Kleine, Gregory A. Keoleian, Jeremy Good, Geoffrey M. Lewis (2017) "Shared autonomous vehicles as a sustainable solution to the last mile problem: A case study of ann arbor-detroit area." SAE International Journal of Passenger Cars-Electronic and Electrical Systems 10.2017-01-1276: 328-336.

14) Charlie Johnston and Jonathan Walker (2017), Peak Car Ownership: The Market Opportunity for Electric Automated Mobility Services, Rocky Mountain Institute (www.rmi.org); at http://bit.ly/2rhJRNi.

15) Prateek Bansal, Kara Kockelman and Amit Singh (2016), “Assessing Public Opinions of and Interest in New Vehicle Technologies: An Austin Perspective”, The

University of Texas at Austin, http://www.caee.utexas.edu/prof/kockelman/public_html/TRB16NewTechsAustin.pdf.

16) Gruel, W., and Stanford, J.M. (2016) “Assessing the Long-Term Effects of Autonomous Vehicles: A Speculative Approach”, Transportation Research Procedia, Vol. 13, pp. 18-29.

17) Mohammad M. Vazifeh, Hongmou Zhang, Paolo Santi and Carlo Ratti (2019), “Optimizing the deployment of electric vehicle charging stations using pervasive mobility data”, Transportation Research Part A: Policy and Practice, Vol 121, 75-91.

18) Jian Ma and Liyan Zhang (2018), “A Deploying method for predicting the size and optimizing the location of an electric vehicle charging stations”, Information (Switzerland), Vol 9, Issue 7.

19) Michael Hyland and Hani Mahmassani (2018), Dynamic autonomous vehicle fleet operations: Optimization-based strategies to assign AVs to immediate traveler demand requests, Transportation Research Part C 92, 278-297.

20) Tesla Website, https://www.tesla.com. Last accessed July 25, 2019.

21) Fred Lambart (2019), “Elon Musk makes incredible claims about Tesla Model 3 longevity, will offer battery module replacement”, https://electrek.co/2019/04/13/tesla-model-3-longevity-claims-elon-musk/. Last accessed May 02, 2019.

22) Autolist.com, https://www.autolist.com/guides/how-long-should-car-last. Last accessed February 7, 2020.

23) Ford Website, https://shop.ford.com/build/focus/#/select/. Last accessed July 25, 2019.

24) American Automobile Association (AAA) (2018), "Your driving costs, How much you are paying to drive".

25) Edmunds.com, https://www.edmunds.com/ford/focus/2018/cost-to-own/#style=401726990. Last accessed February 7, 2020.

26) Zia Wadud (2017), Fully automated vehicles: A cost of ownership analysis to inform early adoption, Transportation Research Part A: Policy and Practice, 101, pp 163-176.

27) Todd Litman (2019), "Autonomous Vehicle Implementation Predictions: Implications for Transport Planning", Victoria Transport Policy Institute.

28) Federal Highway Administration. (2017). 2017 National Household Travel Survey, U.S. Department of Transportation, Washington, DC. Available online: https://nhts.ornl.gov Last accessed Apr 27, 2019.

29) Patrick Sisson (2017), "Uber and Lyft return to Austin: What's changed and why it's important", https://www.curbed.com/2017/6/14/15803138/austin-uber-lyft-transportation-ride-hailing-return. Last accessed May 02, 2019.

30) Meredith Broussard (2018), "The Dirty Truth Coming for Self-Driving Cars: Trash. Odors. Bodily Fluids. Will Autonomous Rideshares be Ready for our Mess?", https://slate.me/2Ls9IrI. Last accessed March 05, 2020.

31) Plugless Power, https://www.pluglesspower.com/learn/tesla-model-s-charging-home-public-autonomously/. Last accessed March 05, 2020.

APPENDIX A

1) Day 0 Simulation – day_0_sim.py

```
1.    import numpy as np
2.    import pandas as pd
3.    from trip_utils import grid_util
4.    from load_data import data_loader
5.    from sim_objects import sav,trip,trip_wait
6.    ## Create virtual Grid
7.
8.    grid = grid_util(5,3,15,0.25)
9.    grid.generate_map_layout()
10.
11.
12.   ## Load O-D Trips and Speed Data
13.   mpr = 5 # 5% MPR
14.   loader = data_loader()
15.   loader.load_trip_data(mpr)
16.   loader.load_speed_data()
17.   # a,b,c = loader.get_dist_trips(1)
18.
19.   ## Initialize charging stations
20.   css = grid.generate_empty_css()
21.   css_copy = grid.generate_empty_css()
22.
23.   # Initialize other hyper params
24.   days = 3
25.   t = 1440 #min
26.   trips = dict()
27.   car_count_dict = dict()
28.   trips_completed = []
29.   cars = []
30.   wait_list = []
31.   car_count = 0
32.   trip_count = 0
33.   wait_count = 0
34.   max_charge = 250
35.
36.   # Day 0 - Simulation
37.   count = 0
38.   for day in range(1,days+1):
39.       print("Day : "+str(day))
40.       for t_step in range(0,t):
41.           ## Status
42.           print("Time : "+ str(t_step))
43.           print("Trips Running : "+str(len(trips)))
44.           print("Cars : "+str(len(cars)))
45.           print("Trips Completed : "+str(len(trips_completed)))
46.           print("Trips in Total: "+str(trip_count))
47.           # New Trips in each step
48.           semi_hr = int(t_step/30)
```

```
            trip_speed_dist,trip_od_dist,trips_in_step =
loader.get_dist_trips(semi_hr)
            # print(trips_in_step)
            for each in range(0,trips_in_step):
                count = count + 1
                # New Trips Generation
                trip_od_zones =
np.random.choice(grid.rel_key.flatten(),1, p=
trip_od_dist.flatten())[0]
                # print("-------")
                # print(trip_od_zones)
                origin_like =
np.random.choice(grid.int_zone_key.flatten(), 1,
p=grid.int_zone_dist.flatten())[0]
                origin = [grid.zones[trip_od_zones[0]][0] * 15 +
origin_like[0], grid.zones[trip_od_zones[0]][1] * 15 +
origin_like[1]]
    #             print(origin_like)
    #             print(origin)
                dest_like =
np.random.choice(grid.int_zone_key.flatten(), 1,
p=grid.int_zone_dist.flatten())[0]
                dest = [grid.zones[trip_od_zones[1]][0] * 15 +
dest_like[0], grid.zones[trip_od_zones[1]][1] * 15 + dest_like[1]]
    #             print(dest_like)
    #             print(dest)
                trip_length = grid.node_distance(origin,dest)
                # Add immediately to wait list
                trip_speed =
trip_speed_dist[trip_od_zones[0]][trip_od_zones[1]]

wait_list.append(trip_wait(wait_count,origin,dest,trip_speed,trip_len
gth,t_step))
                wait_count += 1

                # Process waitlist and assign new trips
                for tw_i, tw in enumerate(wait_list):
                    # Assign by checking charging station
                    origin = tw.origin
                    dest = tw.dest
                    trip_speed = tw.speed
                    trip_length = tw.trip_length
                    tot_dist = grid.charge_distance(origin) +
grid.node_distance(origin,dest) + grid.charge_distance(dest)
                    del_t = t_step - tw.assign_time
                    cs_list = []
                    if del_t >= 2: # 2min minimum waittime
                        if del_t <= 5: # 5min

cs_list.append(grid.nearest_cs_id(origin))
                        else: # search next nerest after 5 min
                            cs_list =
grid.next_nearest_cs_ids(origin) # next charge station
```

```
                        # Check for avaliable cars in all CS
possible
                        for cs_id in cs_list:
                            cs = css[cs_id]
                            car_id_assgn = -1
                            car_i = -1
                            if len(cs.savs) > 0:
                                for i, car_id in
enumerate(cs.savs):
                                    if cars[car_id].on_trip == 0:
                                        car_dist =
grid.node_distance(cars[car_id].location,origin)+
grid.node_distance(origin,dest) + grid.charge_distance(dest)
                                        if car_dist <= tot_dist:
                                            if car_dist <=
cars[car_id].charge: # charge
                                                car_id_assgn =
car_id
                                                car_i = i
                                                break
                                        else:
                                            if i == 0: # assign the
first vehicle
                                                if tot_dist <=
cars[car_id].charge: # charge
                                                    car_id_assgn =
car_id
                                                    car_i = i #no
break
                            if car_id_assgn != -1:
                                old_trip_id =
cars[car_id_assgn].trip_id
                                if old_trip_id != -1:
                                    trips_completed.append(trips[old_trip_id])
                                    del trips[old_trip_id]
                                trips[trip_count] =
trip(trip_count,car_id_assgn,origin,dest,trip_speed,trip_length,t_ste
p,del_t)
                                cars[car_id_assgn].trip_id =
trip_count
                                trip_count = trip_count + 1
                                trip_a = 1
                                del cs.savs[car_i]
                                del wait_list[tw_i]
                                break
                            if del_t>= 10 and car_id_assgn == -1:
                                # Check if charge is sufficient
                                if tot_dist <= max_charge:
                                    car_id_assgn = car_count
                                    cars.append(sav(car_count,cs.id,max_charge,grid.nearest_cs(origin)))
```

```
122.
css_copy[grid.nearest_cs_id(origin)].savs.append(car_count)
123.                                     trips[trip_count] =
trip(trip_count,car_count,origin,dest,trip_speed,trip_length,t_step,d
el_t)
124.                                     cars[car_count].trip_id =
trip_count
125.                                     trip_count = trip_count + 1
126.                                     car_count = car_count + 1
127.                                     trip_a = 1
128.                                     del wait_list[tw_i]
129.                                     break
130.
131.           # Trip starting
132.           # Old trips processing
133.           for tr_id,tr in trips.copy().items():
134.               car_id = tr.sav_id
135.               cars[car_id].speed = tr.trip_speed #
max(tr.trip_speed,2)
136.   #             print(tr.trip_speed)
137.               cs = grid.nearest_cs_id(cars[car_id].location)
138.               if tr.trip_state == 0: # 0 - assigned
139.                   cars[car_id].on_trip = 1
140.                   cars[car_id].on_charging = 0
141.                   tr.start_time = t_step
142.                   tr.trip_state = 1
143.               elif tr.trip_state == 1: #  1 - cs to origin
144.                   cars[car_id].move(tr.origin)
145.                   if cars[car_id].location == tr.origin:
146.                       tr.trip_state = 2
147.               # pickup delay
148.               elif tr.trip_state == 2: # 2 - pickup
149.                   tr.passenger = 1
150.                   tr.trip_state = 3
151.                   tr.pick_time = t_step
152.               elif tr.trip_state == 3: # 3 - origin to
destination
153.                   cars[car_id].move(tr.dest)
154.                   if cars[car_id].location == tr.dest:
155.                       tr.trip_state = 4
156.               #dropoff delay
157.               elif tr.trip_state == 4: # 4 - drop off
158.                   tr.passenger = 0
159.                   tr.drop_time = t_step
160.                   tr.trip_state = 5
161.                   cars[car_id].on_trip = 0
162.                   css[cs].savs.append(car_id)
163.               elif tr.trip_state == 5: # 5 - destination to cs
164.                   cs_loc= grid.nearest_cs(cars[car_id].location)
165.
cars[car_id].move(grid.nearest_cs(cars[car_id].location))
166.                   if cars[car_id].location == cs_loc:
167.                       tr.trip_state = 6
```

```
168.                 elif tr.trip_state == 6: # 6 - docking
169.                     tr.trip_state = 7
170.                     cars[car_id].on_charging = 1
171.                 else:
172.                     trips_completed.append(tr)
173.                     cars[car_id].trip_id = -1
174.                     del trips[tr_id]
175.
176.             # Charging Process inside Charge station
177.             for cs in css:
178.                 if len(cs.savs) > 0:
179.                     for car_id in cs.savs:
180.                         if cars[car_id].charge < max_charge and
cars[car_id].on_charging == 1:
181.                             cars[car_id].charge += 0.75   #60 miles
in 4hrs(240min)
182.                             if cars[car_id].charge> max_charge:
183.                                 cars[car_id].charge = max_charge
184.             car_count_dict[day*24*60 + t_step] = {
185.                 "Day":day,
186.                 "Time":t_step,
187.                 "Trip Count":len(trips),
188.                 "Car Count":len(cars)
189.             }
190.   ## Output files
191.   car_count_df= pd.DataFrame.from_dict(car_count_dict,"index")
192.   car_count_df.to_excel('day_0_results/car_count_thesis.xlsx')
```

2) Day 1 Simulation – day_1_sim.py

```
1. import numpy as np
2. import pandas as pd
3. from trip_utils import grid_util
4. from load_data import data_loader
5. from sim_objects import trip,trip_wait
6. ## Create virtual Grid
7.
8. grid = grid_util(5,3,15,0.25)
9. grid.generate_map_layout()
10.
11.
12.  ## Load O-D Trips and Speed Data
13.  mpr = 5 # 5% MPR
14.  loader = data_loader()
15.  loader.load_trip_data(mpr)
16.  loader.load_speed_data()
17.  # a,b,c = loader.get_dist_trips(1)
18.
19.  ## Initialize charging stations
20.  css = grid.generate_empty_css()
21.  css_copy = grid.generate_empty_css()
```

```
22.
23.  # Initialize other hyper params
24.  ## Initialize charging
25.  max_charge = 250
26.  veh_count_cs = [ 63, 195, 186, 177, 201, 116, 261, 380, 178,
   128, 250, 115,  70, 623, 152 ]
27.  css , cars = grid.generate_filled_css_cars(veh_count_cs,
   max_charge)
28.  #1 [138, 133, 109, 402, 163, 66, 252, 629, 98, 37, 461, 71, 75,
   358, 103]
29.  #2 [ 68, 183, 211, 236, 252, 109, 286, 424, 141,  73, 223,  79,
   99, 557, 154]
30.  #3 [ 63, 195, 186, 177, 201, 116, 261, 380, 178, 128, 250, 114,
   70, 623, 152 ]
31.
32.  t = 1440 #min
33.  days = 5
34.  trips = dict()
35.  wait_list = []
36.  trips_completed = []
37.  trip_count = 0
38.  wait_count = 0
39.  trip_not_found = 0
40.  car_id_store = 1
41.  css_dist= []
42.  car_count_dict = dict()
43.  trip_create_dict = dict()
44.  trip_complete_dict = dict()
45.  car_dict = dict()
46.  trip_not_dict = dict()
47.
48.  # Day 1 - Simulation
49.  count = 0
50.  for day in range(1,days+1):
51.      print("Day : "+str(day))
52.      css_counts  = np.zeros(15)
53.      for car in cars:
54.          cs_id = grid.nearest_cs_id(car.location)
55.          css_counts[cs_id] += 1
56.      css_dist.append(css_counts)
57.      for t_step in range(0,t):
58.          ## Status
59.          print("Time : "+ str(t_step))
60.          print("Trips Running : "+str(len(trips)))
61.          print("Cars : "+str(len(cars)))
62.          print("Trips Completed : "+str(len(trips_completed)))
63.          print("Trips in Total: "+str(trip_count))
64.          # New Trips in each step
65.          semi_hr = int(t_step/30)
66.          trip_speed_dist,trip_od_dist,trips_in_step =
   loader.get_dist_trips(semi_hr)
67.  #          print(trips_in_step)
68.          for each in range(0,trips_in_step):
```

```
                count = count + 1
                #print(css[0].savs)
                # New Trips Generation
                trip_od_zones =
    np.random.choice(grid.rel_key.flatten(),1, p=
    trip_od_dist.flatten())[0]
                found = 0
                while found == 0:
                    origin_like =
    np.random.choice(grid.int_zone_key.flatten(), 1,
    p=grid.int_zone_dist.flatten())[0]
                    origin = [grid.zones[trip_od_zones[0]][0] * 15
    + origin_like[0], grid.zones[trip_od_zones[0]][1] * 15 +
    origin_like[1]]
                    dest_like =
    np.random.choice(grid.int_zone_key.flatten(), 1,
    p=grid.int_zone_dist.flatten())[0]
                    dest = [grid.zones[trip_od_zones[1]][0] * 15 +
    dest_like[0], grid.zones[trip_od_zones[1]][1] * 15 +
    dest_like[1]]
     #                 if origin != dest: ## ensuring minimum trip
    distance
                    found = 1
                # Add immediately to wait list
                trip_speed =
    trip_speed_dist[trip_od_zones[0]][trip_od_zones[1]]
                trip_length = grid.node_distance(origin,dest)

    wait_list.append(trip_wait(wait_count,origin,dest,trip_speed,trip
    _length,t_step))

    #(self,_id,_origin,_dest,_speed,_trip_length,assign_time):

                wait_count += 1

                # Process waitlist and assign new trips
                for tw_i, tw in enumerate(wait_list):
                    # Assign by checking charging station
                    origin = tw.origin
                    dest = tw.dest
                    trip_length = tw.trip_length
                    trip_speed = tw.speed
                    tot_dist = grid.charge_distance(origin) +
    grid.node_distance(origin,dest) + grid.charge_distance(dest)
                    del_t = t_step - tw.assign_time
                    if del_t < 0:
                        del_t = del_t + 1439
                    cs_list = []
    #                 if del_t >=0: # 0min minimum waittime
                    if del_t <= 3: # 3min
                        cs_list.append(grid.nearest_cs_id(origin))
                    elif del_t > 3 and del_t <=8:
```

```
                    cs_list = grid.next_nearest_cs_ids(origin)
  # next charge station
                else: # del_t>= 8
                    trip_not_found += 1
                    print("Trip Cancelled on Day: ", day)
                    del wait_list[tw_i]
                    break
                # Check for avaliable cars in all CS possible
                for cs_id in cs_list:
                    cs = css[cs_id]
                    car_id_assgn = -1
                    car_i = -1
                    if len(cs.savs) > 0:
                        for i, car_id in enumerate(cs.savs):
                            if cars[car_id].on_trip == 0:
                                car_dist =
  grid.node_distance(cars[car_id].location,origin)+
  grid.node_distance(origin,dest) + grid.charge_distance(dest)
                                if car_dist <= tot_dist:
                                    if car_dist <=
  cars[car_id].charge: # charge
                                        car_id_assgn = car_id
                                        car_i = i
                                        tot_dist = car_dist
                                        break
                                else:
                                    if i == 0:
                                        if tot_dist <=
  cars[car_id].charge: # charge
                                            car_id_assgn =
  car_id
                                            car_i = i
                    if car_id_assgn != -1:
                        old_trip_id =
  cars[car_id_assgn].trip_id
                        if old_trip_id != -1:

  trips_completed.append(trips[old_trip_id])
                            trip_complete_dict[trip_count] = {
                                "Trip ID":old_trip_id,
                                "Car ID":
  trips[old_trip_id].sav_id,
                                "Day":day,
                                "Time":t_step,
                                "Wait
  Time":trips[old_trip_id].waited_time,

  "Origin":trips[old_trip_id].origin,

  "Destination":trips[old_trip_id].dest,
                                "Trip
  length":trips[old_trip_id].trip_length,
```

```
                                    "Total
    length":cars[trips[old_trip_id].sav_id].miles
                                }

    cars[trips[old_trip_id].sav_id].miles = 0
                                del trips[old_trip_id]
                            trips[trip_count] =
    trip(trip_count,car_id_assgn,origin,dest,trip_speed,trip_length,t
    _step, del_t)
                            trip_create_dict[trip_count] = {
                                "Trip ID":trip_count,
                                "Car ID": car_id_assgn,
                                "Day":day,
                                "Time":t_step,
                                "Wait Time":del_t,
                                "Origin":origin,
                                "Destination":dest,
                                "Trip length":trip_length,
                            }

    #(self,_id,_sav_id,_origin,_dest,_speed,_trip_length,assign_time)
    :
                            cars[car_id_assgn].trip_id = trip_count
                            trip_count = trip_count + 1
                            trip_a = 1
                            del cs.savs[car_i]
                            del wait_list[tw_i]
                            break


            # Trip starting
            # Old trips processing
            # print("Time step :" + str(t_step))
            for tr_id,tr in trips.copy().items():
                #print("No of trips"+ str(len(trips)))
                car_id = tr.sav_id
                cars[car_id].speed = tr.trip_speed
                cs = grid.nearest_cs_id(cars[car_id].location)
                if tr.trip_state == 0: # 0 - assigned
                    cars[car_id].on_trip = 1
                    cars[car_id].on_charging = 0
                    tr.start_time = t_step
                    tr.trip_state = 1
                elif tr.trip_state == 1: #  1 - cs to origin
                    cars[car_id].move(tr.origin)
                    if cars[car_id].location == tr.origin:
                        tr.trip_state = 2
                # pickup delay
                elif tr.trip_state == 2: # 2 - pickup
                    tr.passenger = 1
                    tr.trip_state = 3
                    tr.pick_time = t_step
```

```
            elif tr.trip_state == 3: # 3 - origin to
   destination
                #step =
   step_closer(cars[car_id].location,tr.dest)
                cars[car_id].move(tr.dest)
                if cars[car_id].location == tr.dest:
                    tr.trip_state = 4
            #dropoff delay
            elif tr.trip_state == 4: # 4 - drop off
                tr.passenger = 0
                tr.drop_time = t_step
                tr.trip_state = 5
                cars[car_id].on_trip = 0
                css[cs].savs.append(car_id)
            elif tr.trip_state == 5: # 5 - destination to cs
                cs_loc= grid.nearest_cs(cars[car_id].location)

   cars[car_id].move(grid.nearest_cs(cars[car_id].location))
                #step =
   step_closer(cars[car_id].location,nearest_cs(cars[car_id].locatio
   n))
                if cars[car_id].location == cs_loc:
                    tr.trip_state = 6
            elif tr.trip_state == 6: # 6 - docking
                tr.trip_state = 7
                cars[car_id].on_charging = 1
            else:
                trips_completed.append(tr)
                cars[car_id].trip_id = -1
                trip_complete_dict[trip_count] = {
                    "Trip ID":tr_id,
                    "Car ID": tr.sav_id,
                    "Day":day,
                    "Time":t_step,
                    "Wait Time":tr.waited_time,
                    "Origin":tr.origin,
                    "Destination":tr.dest,
                    "Trip length":tr.trip_length,
                    "Total length":cars[car_id].miles
                }
                cars[car_id].miles = 0
                del trips[tr_id]
        # Charging Process inside Charge station
        for cs in css:
            if len(cs.savs) > 0:
                for car_id in cs.savs:
                    if cars[car_id].charge < max_charge and
   cars[car_id].on_charging == 1:
                        cars[car_id].charge += 0.75 # 60 miles
   in 4hrs(240min)
                        if cars[car_id].charge> max_charge:
                            cars[car_id].charge = max_charge
        # Store a vehicle path/2
```

```
            car_dict[(day-1)*60*24+t_step] = {
                "Car ID": car_id_store,
                "Day": day,
                "Time": t_step,
                "On Trip": cars[car_id_store].on_trip,
                "Charge": cars[car_id_store].charge,
                "Location": cars[car_id_store].location
            }
            car_count_dict[day*24*60 + t_step] = {
                "Day":day,
                "Time":t_step,
                "Trip Count":len(trips),
                "Trips Cancelled":trip_not_found
            }
print(count)
# Output files
trip_df_exl = pd.DataFrame.from_dict(trip_create_dict,"index")
trip_df_exl.to_excel('day_1_results/create_trips_thesis.xlsx',e
   ngine='xlsxwriter')
trip1_df_exl =
   pd.DataFrame.from_dict(trip_complete_dict,"index")
trip1_df_exl.to_excel('day_1_results/completed_trips_thesis.xls
   x',engine='xlsxwriter')
car_df_exl= pd.DataFrame.from_dict(car_dict,"index")
car_df_exl.to_excel('day_1_results/cars_thesis1.xlsx',engine='x
   lsxwriter')
car_count_df = pd.DataFrame.from_dict(car_count_dict,"index")
car_count_df.to_excel('day_1_results/car_counts1.xlsx',engine='
   xlsxwriter')
```

3) Trips Utils – trip_utils.py

```
import numpy as np
from sim_objects import charge_station,sav

class grid_util:
    def __init__(self,zone_w,zone_h,zone_res,node_dist):
        self.zone_w = zone_w
        self.zone_h = zone_h
        self.zone_res = zone_res
        self.zone_mid = int(zone_res / 2)
          self.node_dist = node_dist

     def generate_map_layout(self):
         ## Initialize all zones and make relational keys
         self.zones = dict()
         c = 0
         for j in range(0,self.zone_w):
             for i in range(0,self.zone_h):
                     self.zones[c] = [i,j]
                     c+=1
```

```
20.          # Zone size - 15 x 15
21.          self.int_zone_dist =
   np.zeros([self.zone_res,self.zone_res])
22.          for i in range(0,self.zone_res):
23.              for j in range(0,self.zone_res):
24.                  self.int_zone_dist[i][j] = 1/(self.zone_res
   *self.zone_res) ## equal probablity of selecting any node
25.          self.int_zone_key =
   np.zeros([self.zone_res,self.zone_res],dtype=object)
26.          for i in range(0,self.zone_res):
27.              for j in range(0,self.zone_res):
28.                  self.int_zone_key[i][j] = [i,j] ## Key for the
   probability
29.          self.rel_key =
   np.empty([self.zone_res,self.zone_res],dtype=object)
30.          for i in range(0,self.zone_res):
31.              for j in range(0,self.zone_res):
32.                  self.rel_key[i][j] = [i,j] ##
33.
34.      def node_distance(self,nd1,nd2):
35.          dist = (abs(nd1[0] - nd2[0]) + abs(nd1[1] - nd2[1])) *
   self.node_dist # 0.25 mile is the internodal distance
36.          return dist
37.
38.      def charge_distance(self,nd):
39.          x = nd[0]% self.zone_res
40.          y = nd[1]% self.zone_res
41.          dist = (abs(x - self.zone_mid) + abs(y - self.zone_mid)) *
   self.node_dist # 0.25 mile is the internodal distance
42.          return dist
43.
44.      def nearest_cs(self,nd):
45.          x = int(nd[0]/self.zone_res)
46.          y = int(nd[1]/self.zone_res)
47.          return
   [x*self.zone_res+self.zone_mid,y*self.zone_res+self.zone_mid]
48.
49.      def nearest_cs_id(self,nd):
50.          x = int(nd[0]/self.zone_res)
51.          y = int(nd[1]/self.zone_res)
52.          return (x + y*self.zone_h)
53.
54.      def next_nearest_cs_ids(self,nd):
55.          x = int(nd[0]/self.zone_res)
56.          y = int(nd[1]/self.zone_res)
57.          l = []
58.          if x-1 >= 0:
59.              l.append((x-1)+y*self.zone_h)
60.          if x+1 <= self.zone_h - 1:
61.              l.append((x+1)+y*self.zone_h)
62.          if y-1 >= 0:
63.              l.append(x+(y-1)*self.zone_h)
64.          if y+1 <= self.zone_h + 1:
```

```
                l.append(x+(y+1)*self.zone_h)
            return l

        def generate_empty_css(self):
            css = []
            count = 0
            for j in range(0,self.zone_w):
                for i in range(0,self.zone_h):
                    css.append(charge_station(count,[i*7+3,j*7+3]))
                    count = count +1
            return css

        def generate_filled_css_cars(self,veh_count_cs,max_charge):
            css = []
            cars = []
            cs_id = 0
            car_id = 0
            for j in range(0,5):
                for i in range(0,3):
                    css.append(charge_station(cs_id,[i*15+7,j*15+7]))
                    for k in range(0,veh_count_cs[cs_id]):
    cars.append(sav(car_id,cs_id,max_charge,[i*15+7,j*15+7]))
                        css[cs_id].savs.append(car_id)
                        car_id += 1
                    cs_id += 1
            return css,cars
```

4) Data Loader – load_data.py

```
import numpy as np

class data_loader:
    def __init__(self):
        ## Time zones
        self.am_time = 3 * 60 # 6 -9
        self.md_time = 6.5 * 60 # 9 - 15:30
        self.pm_time = 3 * 60 # 15:30 - 18:30
        self.nt_time = 11.5 * 60 # 18:30 - 6


    def load_trip_data(self,mpr):
        dist_am = np.load('dist_am.npy')
        dist_md = np.load('dist_md.npy')
        dist_pm = np.load('dist_pm.npy')
        dist_nt = np.load("dist_nt.npy")
        ## MPR 5%
        am_trip = dist_am * (mpr/100)
        md_trip = dist_md * (mpr/100)
        nt_trip = dist_nt * (mpr/100)
        pm_trip = dist_pm * (mpr/100)
```

```
            ## Convert to Probability Distribution
            self.am_trip_dist = am_trip/am_trip.sum()
            self.md_trip_dist = md_trip/md_trip.sum()
            self.nt_trip_dist = nt_trip/nt_trip.sum()
            self.pm_trip_dist = pm_trip/pm_trip.sum()

            ## Trip dist
            am_total = am_trip.sum()
            md_total = md_trip.sum()
            pm_total = pm_trip.sum()
            nt_total = nt_trip.sum()

            ## Trip per min
            self.am_trip_min = int(am_total/self.am_time)
            self.md_trip_min = int(md_total/self.md_time)
            self.pm_trip_min = int(pm_total/self.pm_time)
            self.nt_trip_min = int(nt_total/self.nt_time)

        def load_speed_data(self):
            ## Speed Data Probability Distribution
            self.dist_speed_am =
    np.load('dist_speed_am.npy').astype(int)
            self.dist_speed_pm =
    np.load('dist_speed_pm.npy').astype(int)
            self.dist_speed_md =
    np.load('dist_speed_md.npy').astype(int)
            self.dist_speed_nt =
    np.load('dist_speed_nt.npy').astype(int)

        def get_dist_trips(self,semi_hr):
            if semi_hr >= 12 and semi_hr <18:
                print("AM")
                trip_speed_dist = self.dist_speed_am
                trip_od_dist =  self.am_trip_dist
                trips_in_step = self.am_trip_min
            elif semi_hr >= 18 and semi_hr < 31:
                print("MD")
                trip_speed_dist = self.dist_speed_md
                trip_od_dist =  self.md_trip_dist
                trips_in_step = self.md_trip_min
            elif semi_hr >= 31 and semi_hr < 37:
                print("PM")
                trip_speed_dist = self.dist_speed_pm
                trip_od_dist =  self.pm_trip_dist
                trips_in_step = self.pm_trip_min
            else:
                print("NT")
                trip_speed_dist = self.dist_speed_nt
                trip_od_dist = self. nt_trip_dist
                trips_in_step = self.nt_trip_min

            return trip_speed_dist,trip_od_dist,trips_in_step
```

5) Simulation Objects – sim_objects.py

```
1. class sav:
2.     def __init__(self,_id,_cs,_charge,_location):
3.         self.id = _id
4.         self.cs_id = _cs
5.         self.charge = _charge
6.         self.location = _location
7.         self.on_charging = 0
8.         self.miles = 0
9.         self.total_miles = 0
10.          self.speed = 1 # 1 node / min = 0.25 mile/min = 15 mph
11.          # 2 - 30 mph
12.          # 3 - 45 mph
13.          # 4 - 60 mph
14.          self.on_trip = 0
15.          self.trip_id = -1
16.
17.      def move(self,dst):
18.          count = 0
19.          while count < self.speed:
20.              ori = self.location
21.              if dst[0]-ori[0] != 0:
22.                  self.charge -= 1 * 0.25
23.                  self.miles += 1 * 0.25
24.                  self.total_miles += 1 * 0.25
25.                  if dst[0] - ori[0] > 0:
26.                      step = [ori[0]+1,ori[1]]
27.                  else:
28.                      step = [ori[0]-1,ori[1]]
29.              elif dst[1]-ori[1] != 0:
30.                  self.charge -= 1 * 0.25
31.                  self.miles += 1 * 0.25
32.                  self.total_miles += 1 * 0.25
33.                  if dst[1] - ori[1] > 0:
34.                      step = [ori[0],ori[1]+1]
35.                  else:
36.                      step = [ori[0],ori[1]-1]
37.              else:
38.                  step = [ori[0],ori[1]]
39.                  break
40.              self.location = step
41.              count +=1
42.
43.
44.  class charge_station:
45.      def __init__(self,_id,_location):
46.          self.id = _id
47.          self.location = _location
48.          self.savs = []
49.          self.max_cap = 20
50.          self.current_cap = 0
```

```
51.
52.  class trip:
53.      def
   __init__(self,_id,_sav_id,_origin,_dest,_speed,_trip_length,assign_t
   ime,del_t):
54.          self.id = _id
55.          self.sav_id = _sav_id
56.          self.origin = _origin
57.          self.dest = _dest
58.          self.trip_speed = _speed
59.          self.trip_length = _trip_length
60.          self.assign_time = assign_time
61.          self.waited_time = del_t
62.          self.start_time = -1
63.          self.pick_time = -1
64.          self.drop_time =-1
65.          #self.end_time = -1
66.          self.trip_state = 0
67.      # 0 assigned, 1 - cs to origin, 2 - pickup, 3 - origin to
   destination, 4 drop off, 5-destination to cs, 6 docking, 7 complete
68.          self.passenger = 0
69.
70.  class trip_wait:
71.      def
   __init__(self,_id,_origin,_dest,_speed,_trip_length,assign_time):
72.          self.id = _id
73.          self.origin = _origin
74.          self.dest = _dest
75.          self.speed = _speed
76.          self.trip_length = _trip_length
77.          self.assign_time = assign_time
```