EVALUATING THE SUSTAINABILITY IMPACTS OF INTELLIGENT CARPOOLING SYSTEMS FOR SOV COMMUTERS IN THE ATLANTA REGION

A Dissertation
Presented to
The Academic Faculty

by

Diyi Liu

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Civil & Environmental Engineering

Georgia Institute of Technology
December, 2020

EVALUATING THE SUSTAINABILITY IMPACTS OF INTELLIGENT CARPOOLING SYSTEMS FOR SOV COMMUTERS IN THE ATLANTA REGION

Approved by:

Dr. Randall Guensler, Advisor
School of Civil and Environmental Engineering
Georgia Institute of Technology

Dr. Michael Rodgers
School of Civil and Environmental Engineering
Georgia Institute of Technology

Dr. Haobing Liu
School of Civil, Construction and Environmental Engineering
University of New Mexico

Date Approved: [December 3rd, 2020]

For my grandpa, your legends will never be forgotten

ACKNOWLEDGEMENTS

I would like to especially thank all of my colleagues, without whose guidance and support I would not finish this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

ARC	Atlanta Regional Commission
ABM	Activity-based (Travel Demand) Model
TAZ	Traffic Analysis Zone
SOV	Single-Occupancy Vehicle
LP	Linear Programming
ILP	Integer Linear Programming
SN	Shareability Network

SUMMARY

Community-based carpooling has more potential to help alleviate traffic congestion and reduce energy use during peak hours than ride-hailing services, such as Uber or Lyft, because community-based carpooling avoids deadheading operations. However, community-based carpooling is not fully exploited due to communication, demographic, and economic barriers. This thesis proposes a top-down computation framework to estimate the potential market-share of community-based carpooling, given the outputs of activity-based travel demand models. Given disaggregate records of commute trips, the framework tries to estimate a reasonable percentage/number of trips among commuters in single-occupancy vehicles (SOV) that can carpool together, considering spatiotemporal constraints of their trips.

The framework consists of two major procedures: (1) trip clustering; and (2) trip optimization. The framework tackles the problems associated with using large amounts of data (for example, the Atlanta travel demand model predicts more than 19 million vehicle trips per day) by following "split-apply-combine" procedures. A number of tricks and technologies (e.g., pre-computing, databases, concurrency, etc.) are employed to make the mass computing tasks solvable in a personal laptop in a reasonable time.

Two different methods are established to solve the carpooling optimization problem. One method is based on the bipartite algorithm, while the other uses integer linear programming. The linear programming method estimates both the *systemic optimal performance* in terms of saving the most vehicular travel mileage, while the bipartite-based algorithm estimates one *Pareto optimal performance* of such system that pairs the greatest number of carpool members (i.e., maximum number of travelers that can use the system) given acceptable (defined by the user) reroute cost and travel delays. The performance of these two methods are carefully compared.

A set of experiments are run to evaluate the carpooling potentials among single-occupancy vehicles based on the output of activity-based model's (ARC ABM) home-to-work single-occupancy vehicle (SOV) trips that can be paired together towards designated regional employment centers. The experiment showed that under strict assumptions, an upper bound of around 13.6% of such trips can be carpooled together. The distribution of these trips over space, time, and travel network are thoroughly discussed. The results are promising in terms of finding carpooling and decreasing total vehicle mileage. Moreover, the framework is flexible enough with the potential to act as a simulation testbed, to optimize vehicular operations, and to match potential carpool partners in real-time.

CHAPTER 1. INTRODUCTION

1.1 Research Goal

The research is inspired by the following question: Given reasonable spatiotemporal limitations such as additional travel time for passenger pickup, what percentage of commute SOV commute trips can carpool together during the morning commute period? In addition, the research team tries to integrate the following into the optimization process:

1. Benefits in reducing traffic system's congestion burden during peak hours when applying such a carpool system;
2. Similarly, the sustainability effects (e.g., in decreasing the number in carbon footprints).
3. Redundancy and reliability of community-based carpooling system itself.
4. Friendliness and attractiveness of such system to its users.

The research team developed an analytical framework to assess the issues outlined above. Under this analysis framework, a set of different technologies and methods are developed. By using the trip outputs generated from the activity-based travel demand model (ABM), the framework can find a specific optimal carpool assignment plan, which is considered to be the upper-bound potential of the carpooling system.

In this case study, commute trip information is generated by Atlanta Regional Commission's (ARC's) ABM, one of the most advanced travel demand models in the country (Davidson, et al., 2010). The ABM uses the Coordinated Travel – Regional

Activity Modeling Platform (CT-RAMP) platform as operationalized in the Citilabs® Cube model (Davidson, et al., 2010). CT-RAMP first generates a regional population of synthetic households using the PopSynIII model. Each synthetic household is assigned demographic characteristics (e.g., number of residents, household income, vehicle ownership, etc.) and is populated with synthetic persons (specified by such factors as gender, age group, employment status, etc.) to match Census and other demographic characteristics (Davidson, et al., 2010). The ABM uses these household and person parameters to predict each household's and individual's travel activities in the form of shared and individual tours and trips. For each trip, the output trip table contains detailed information such as departure time (in one-half-hour time bins), departure coordinates, trip purpose, travel mode, and specific members of the household sharing the trip. The Georgia Tech team has also implemented a new path retention feature (Zhao, et al., submitted) for ABM that allows researchers to output the trip paths (each link traversed by each trip from origin to destination) created during the Frank-Wolfe traffic assignment algorithms, which is very useful in further spatial and temporal travel analysis.

1.2 Literature Review

1.2.1 Overview

This chapter presents literature review relevant to the modeling of the carpooling/ridesharing/dial-a-ride problem as well as issues in carpooling practices. This study, like many carpooling papers, discussed a Daily Carpooling Problem (DCPP). According to (Calvo RW, et al., 2004), DCPP is a specific case of routing problem with pickup and deliverables and time window. (Calvo RW, et al., 2004) viewed it as a special case of Dial-a-Ride problem (DARP) and heuristics are applied due to the complexity of the problem. In this project, the problem is also treated as DCPP and both heuristics and precise integer linear programming are applied to get a good solution. The project emphasizes on integrating the carpooling problem to the existing agent-based traffic demand models, thus provides a testbed for analyzing futuristic carpooling/vanpooling scenarios.

The project focuses on community-based/home-based carpooling, which is less discussed than the normal work-based/employment-based case where carpooling happens among employees in one company. In practice, the work-based carpooling problem can usually narrow down the problem size from a few hundred to a few thousand trips, considering the number of carpool-able employees within one employer like hospitals or schools or factories are no more that amount. The project also looks at pairing workers from different companies but sharing approximate origin/destination together. This is future-oriented considering involvement of autonomous vehicle, internet-of-things, etc. In other words, this model focuses on upper-bound capability of the carpooling system considering only system efficiency but without information frictions.

There are two different ways of perceiving the carpooling problem. One method, named Traffic Network Model, views the carpooling problem as clustering trips sharing similar trajectory in spatiotemporal dimensions. The other method is to build up a graph named Shareability Network, in which nodes represent traveler with or without role assigned and links represent their likelihood/utilities/cost of carpooling. Both ideas are borrowed and implemented in the proposed analysis framework. Traffic network model is used as clustering trips and narrowing down searching space of carpools while shareability networks are used as precisely model proposed carpool schemes between different travelers. In other words, this framework generates both carpool assignments with role and operation rules at once.

1.2.2 Traffic Network Model

Problems like carpooling can be viewed as clustering trips sharing similar portions of spatiotemporal trajectories. To do this, a certain trip can be described as a sequence of spatiotemporal recordings. If all trips are known, algorithms can be devised to match demand with supply in the travel system. Cruz, et al. (2015) proposed a method of clustering trips into groups by their trip trajectories based on the Optics algorithm. A more complex TNM integrates temporal attributes into the graph, either by setting dynamic link weights or expand the graph along time axis to form a directed acyclic graph (DAG). For example, Jamal, et al. (2017) implemented the two versions of graphs to integrate temporal dependencies and integrate transit operations into the carpooling problem. Dijkstra family algorithms are used in either of the two networks to find shortest paths. Carpool-matching models based on TNM focus on finding matches for individual trips, which emphasize on a detailed operation problem with a small data size. The algorithms like Dijkstra used in these problems are computationally intensive and it suffers computational issues as problem size grows.

1.2.3 Shareability Networks

The other type of graph model in carpool modeling is the shareability network model (SNM), a notion advertised by Santi P, et al. (2014). Unlike traffic network models, a SNM denotes each passenger/entity as a node/vertex and each candidate carpooling match as a link that connects two points. The weight/distance of each link reflects the likelihood of two corresponding parties denoted by the link's two end nodes. The objective is to find the maximum number of links without two links sharing connecting one node (i.e., conflicts that two persons perform two roles). There are three major types of models to model the problem: bipartite model, monopartite model, and integer programming model.

Several efforts have applied these algorithms in carpool matching by using graph matching algorithms. For instance, Zhang, et al. (2019) modeled the preference-matching problem in ridesharing as a monopartite problem. They assume people can be grouped into several different social categories and each group has some preferences to interact/carpool between groups. Different scenarios are assumed for the interaction behavior within or between groups and the algorithm provides results for trip matching in both number and quality. The claim is that preference-based matching provides closer matches than efficiency-based algorithms. Qi, et al. (2016) developed a graph similar to monopartite graph but modeling the problem as finding complete subgraphs for multiple parties pairing.

It is obvious that the optimization problem of shareability networks can be generalized as (integer) linear programming. Alonso-Mora, et al. (2017) designed a new real time algorithm for routing and ridesharing to reach near optimal outcomes. Hsieh, et al. (2018) formulated the carpooling problem as an integer programming problem and develop variants of Derivative Evolution algorithm to solve it. They resolved the carpooling problem by decomposing the analysis into two procedures: the route-planning phase and carpooler-matching phase.

1.2.4 Adopted Methodology

In this NCST project, two methods are proposed in formulating carpooling problem: (1) formulate a bipartite model; (2) a pure integer linear programming formulation. Both methods are able to solve the optimal problem but with different objective functions. The first bipartite model leads to a solution with maximum number of carpooling pairs. The linear programming formulation can have flexible constraints, and the most basic target is to minimizing total vehicular travel time over the system, which guarantees a global systemic optimal value.

One advantages of the proposed method is the concurrent generation of both the pairing plan and the specific routing plan. Given the outputs from the activity-based travel demand model, analysis starts with the individual commute trips, with routes represented as shortest path between the traveler’s OD pair. No matter whether the candidate is assigned to join a carpool trip, the path is recorded, which gives us method to thoroughly compare and discuss the impacts of the system.

CHAPTER 2. ANALYSIS FRAMEWORK

Given population and their traveling data (e.g., ARC ABM outputs), a computational framework is designed to estimate the number of carpool-able trips, as shown in Figure 1. This framework consists of two key modules: (1) trip clustering module; (2) trip optimization module.

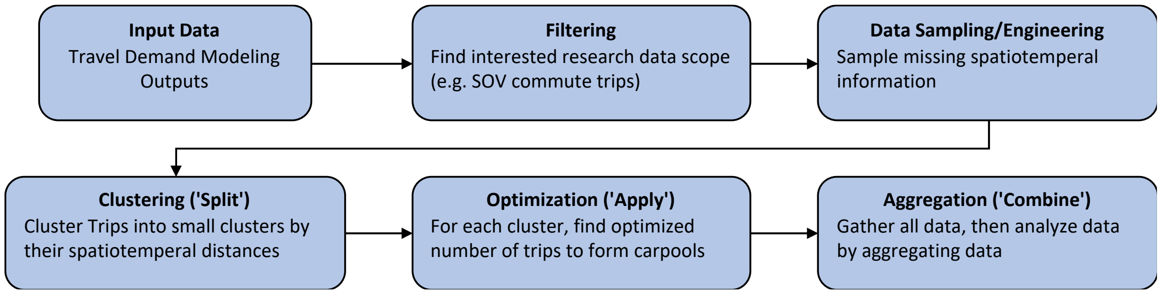

Input Data
Travel Demand Modeling Outputs
Filtering
Find interested research data scope (e.g. SOV commute trips)
Data Sampling/Engineering
Sample missing spatiotemperal information
Clustering ('Split')
Cluster Trips into small clusters by their spatiotemperal distances
Optimization ('Apply')
For each cluster, find optimized number of trips to form carpools
Aggregation ('Combine')
Gather all data, then analyze data by aggregating data

Figure 1 Proposed Carpooling Analysis Framework

The proposed framework is simple and interpretable since it aligns with the well-known idea of "split-apply-combine" pattern in data science/data engineering community. To be specific, the "split" step refers to splitting a problem with big data size into the same problem but with many smaller groups with smaller data size. Since the efficiency of many computation problem is very sensitive to data sizes, one can gain speed and become more convenient in by solving many problems with small data size instead of solving a large problem at once. The "apply" step refers to apply a function to solve the problem for each individual group. The "combine" step is to gather outputs of every separate group and combine outputs into a single format as the solution to the whole problem. By splitting problem with large amount data into the same problem but with smaller data sizes, the problem can be more efficiently solved by the nature of algorithm’s inefficiency, and the support of multiprocessing/parallel computing techniques.

The first step of the proposed method is to filter out useless trips and cluster useful trips into many smaller groups. Only trips fall into the same group is considered to be paired together. The second step is to find all possible carpool-able combinations among travelers. The last step is to gather the solutions for each individual group thus form the solution to the whole problem.

2.1 Preparation Work (Data Filtering and Data Engineering)

2.1.1 Study scope

In the current phase, the researcher focuses on estimating the number within the most appropriate population to study. The group focuses on the driving alone commuters (i.e., single-occupancy vehicles). The researcher further limits the destination of the trips towards so-called employment centers identified by ARC, which usually causes congestion by importing inbound traffic in morning peak hours and outbound traffic during evening peak hours. Trips are further limited to commuting trip purpose (from work to home), which means the study scope’s trips are primarily in the morning peak period. However, many trips will still happen during other time of the day. One example of selecting data of interest is shown in Table 2.

2.1.2 Regenerate precise trip information

Although ARC ABM provides individual-wise travel outputs, there is a gap in preciseness between ABM ARC’s travel demand modeling trip outputs and the demand for carpooling operation research. There are two pieces of information that are not precise: temporal, spatial, and path preciseness. This section discussed those limitations and presents the fixes employed by the researchers.

Temporal issue: ABM trip outputs only aggregate travel depart time to 30-minute time. The 30-minute time-bin is too wide for precise analysis of carpooling. To solve this issue, a spline curve is fit to the departure time for all trips of interest. Then, trips are sampled to a precise minute using the results in the fitted spline curve for the 30-minute time bin into which the trips fall.

Spatial issue: Just like temporal issue, the ABM trip outputs is not precise about the origin/destination location of the trip. The trip only gives the traffic analysis zone the trip starts/ends without further notifying the longitude latitude information of the trip. In fact, all trips are assumed to depart/arrive from the TAZ centroids. To solve this problem, the researcher randomly sampled 100 locations within each traffic analysis zone and randomly sample a coordinate location to the traveler's origin/destination location.

Path retention issue: The path retention provided by ARC ABM only gives trip outputs between TAZ centroids. Carpooling trips sharing similar origins will be clustered into one TAZ centroid and the reroute cost for carpooling might be underestimated. Thus, to build a platform capable of analyzing the costs of carpooling, paths must be generated for the OD trips. The researcher developed a new two-stage method in estimating reasonable traveling paths with very low computational costs. Furthermore, the tool is further improved by enabling the capacity of dynamically generate carpool plans. The methods used to generate data will be discussed in the methodology section.

2.2 Trip Clustering Module

After establishing the scope of the study, the next step is to find maximum number of carpooling trips for a set of reasonable constraints. However, since such optimization problem is NP-complete, trips must first be clustered into smaller groups to enhance computational efficiency. Trips are clustered based on their vicinity in spatiotemporal dimensions. Only trips falling into a same cluster can be considered carpool together. Trips in two separate clusters cannot be carpooled together since the gap in locations.

For this framework, trips are clustered based on their proximity in origin location and departure time. This clustering assumption is accurate for this problem since the paper focuses on community-based carpooling, meaning that trips are carpool-able only if they share similar origin locations and depart time. The problem of whether they really fit to form a carpool will be analyzed in the trip optimization step.

2.3 Trip Optimization Module

After target travelers/trips are clustered based on their spatiotemporal proximity, the system only needs to optimize results within each cluster. After finding optimal results within each trip cluster, we can easily aggregate results for all sub-problems to get the whole picture of the problem.

Two algorithms are proposed to answer the optimization problem. One algorithm uses Bipartite algorithm plus heuristics rules, the other algorithm tries to formulate linear programming optimization problem and use LP solver to get optimal results.

Assuming two travelers A and B are being considered for pairing in a carpool trip, there are four potential operation possibilities (this research only considers the first three options)

- They both continue to drive alone from home to work (default operation).
- If they decide to carpool together, usually the driver picks up and drops off the passenger at the passenger's original and destination location:
    - One scenario is that A is the driver and B is the passenger
    - Another scenario is that A is the passenger and B is the driver
- The last scenario is that A and B both drive to a mid-point (e.g., a park and ride facility) and then carpool together

# CHAPTER 3.	METHODOLOGY

This chapter first discusses the supportive data engineering work to perform carpool matching. Then, the trip clustering methods are discussed. Next, the concept of feasibility matrix is discussed to model the problem. Finally, two different methodologies getting optimal results are discussed.

3.1 Data Processing

3.1.1 Resampling departure/arrival time

The original trip outputs only contain information for 30-minute time bins for departure time (e.g., 6:30-7:00 AM). However, carpool pairing will require estimation of departure time by minute. Instead of assuming the trip uniformly distributed over a 30-minute time bin, the following three steps are undertaken for sampling: (1 plot a histogram of trips depart time counts over 48 30-minute time-bins during the analysis day; (2 fit a spline curve over the histogram; (3 for each trip, sampling a minute out of the 30 minutes as the trip's departure time, assuming the trip must happen in the 30-minute time bin and the departure time follows the probability distribution proportional to that of the spline curve within each 30-minute time bin. For example, Figure 2 shows the trip departure histogram over time by each 30-minute bin, with a red curve as spline curve fitted to the histogram.

Because the ABM outputs provide a precise traveling duration for each trip, the arrival time is the departure time plus trip duration. Another important element in practice is that the trip's travel time, for which the Georgia Tech shortest-path finder (i.e., RoadwaySim) based upon dynamic traffic assignment, is substituted for the ABM output (Frank-Wolfe Algorithms). This is because all carpool operations in later steps are also estimated by RoadwaySim, we need to provide a uniform way of assessing travel time and the travel paths.

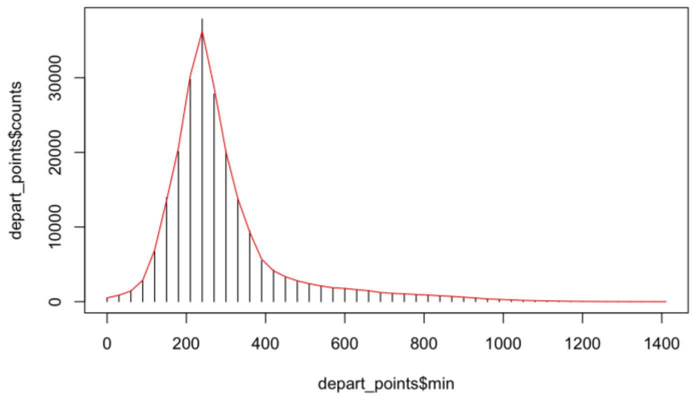


Figure 2 Histogram of departure time over every 30-minute time bins with its spline curved

3.1.2 Resampling origin/destination coordinates

ABM ARC only gives the TAZ number the trip departs from/arrives at, and it fails to provide a precise geometry location of the trip (e.g., longitude/latitude), which is bad for measuring microscopic operations. Considering the tradeoffs between computation complexity and precision, the following sampling plan is employed in practice: (1) Randomly sampling 100 geometry coordinates/points within each TAZ parcel, and index the 100 locations using number from 1-100; (2) For any trip using the TAZ as origin/destination, randomly assign an index (from 1-100) for one of the precise coordinates.

The original ABM's path retention outputs provide the specific paths between TAZ centroids. Each TAZ centroid has at least one connector to the traffic network. In ABM, all trips depart from a same TAZ starts at the same location and usually uses a same connector to the nearest network. This simplification somewhat undermines the ability to analyze carpool operations. To solve this issue, we compute all trip trajectories using RoadwaySim, which will be thoroughly discussed in the next section. Briefly speaking, we first using RoadwaySim to compute the shortest paths between any origin-destination TAZ centroid. Then we design an algorithm to approximate the shortest paths between precise origin-destination coordinates.

3.2 Shortest-path Estimation

3.2.1 Two-stage Shortest Path Estimation

Finding shortest path is computationally expensive using Dijkstra-family algorithm, which is especially true considering that the system is estimating millions of trips. A two-stage process for finding shortest path between any coordinates is proposed below.

In the first stage, the system enumerates all possible TAZ origin-destination (OD) combinations. For each pair of TAZ centroids, the shortest path is estimated using RoadwaySim. Thus, we have the shortest path between any possible pair of TAZs. This pre-computed data is stored in a database for fast query efficiency.

After the first step, we only know the shortest path between TAZ centroids, not between precise OD coordinates. The second step is to use a pre-computed shortest path between TAZ centroids to quickly approximate the shortest path between any coordinates. The procedure steps of this approximation algorithm are listed as the following table:

Table 1. Procedures of the Two-step Shortest Path Algorithms

Stage	**Description**
1.	For all combination of origins and destinations, compute the shortest link paths between the pair and store the results in a database for fast query.
2.	Given any pair of origin & destination coordinates: (1) Find the origin/destination the trip falls into. (2) Check the shortest paths between OD TAZ centroids from the database. (3) (Temporally) Set the weights of all links in the shortest paths to a very small positive number. (4) Rerun the Dijkstra's algorithm to speed-up the computation. (5) Reset the original weights of all travel links. The result generated by step 4 is treated as the updated shortest path. This path approximates the real shortest path.

Because Dijkstra's algorithm searches a priority queue, by setting costs of pre-computed shortest paths to zero, we prioritize the costs of the trips in the search frontier, making it much faster to find the destination node in the network. While the speed gain is easy to understand, it is not straight forward to show the optimality for the two-stage procedure. By making the assumption that stage 1's path contains the shortest path between the corresponding TAZs cycling roads, we can prove that the two-stage algorithm generates the sub-optimal path with small constant. This claim (i.e., proposition 1) is discussed in the Appendix. In practice, the algorithm generates high quality shortest paths most of the time.

3.2.2 Dynamic Shortest-path estimation (CarpoolSim)

The same shortest path computation method can be extended to estimate travel paths for different trip segments of the carpooling trips. For example, assume that A and B are considered to carpool together for morning commute. The first scheme is to do nothing but let them drive alone to work as they did before. A second scheme is to assign A as the role of driver and B as passenger. A would depart from home and then pick-up B and drop off B before A arrives at his/her destination. Similarly, the second scheme is to assign B to become the driver and assign A as the passenger. In this way, we enumerate three possible operation schemes considered by the carpooling system.

Figure 3 demonstrates the three basic carpooling scenarios as discussed above. The first scheme is both parties choose to drive alone (blue and red line). The second plan is to assign party 2 as the driver and form carpool trip with person 1. The third plan is to assign party 1 as the driver and form carpool trip with person 2. The Georgia Tech team proposed a computing framework called CarpoolSim with the capability to analyze the shortest paths for complex traveling cases like carpooling. To model the scheme, we only need to run

CarpoolSim once for every trip segment. The time/distance cost of each trip segment is computed and used as inputs to build up an optimization model.

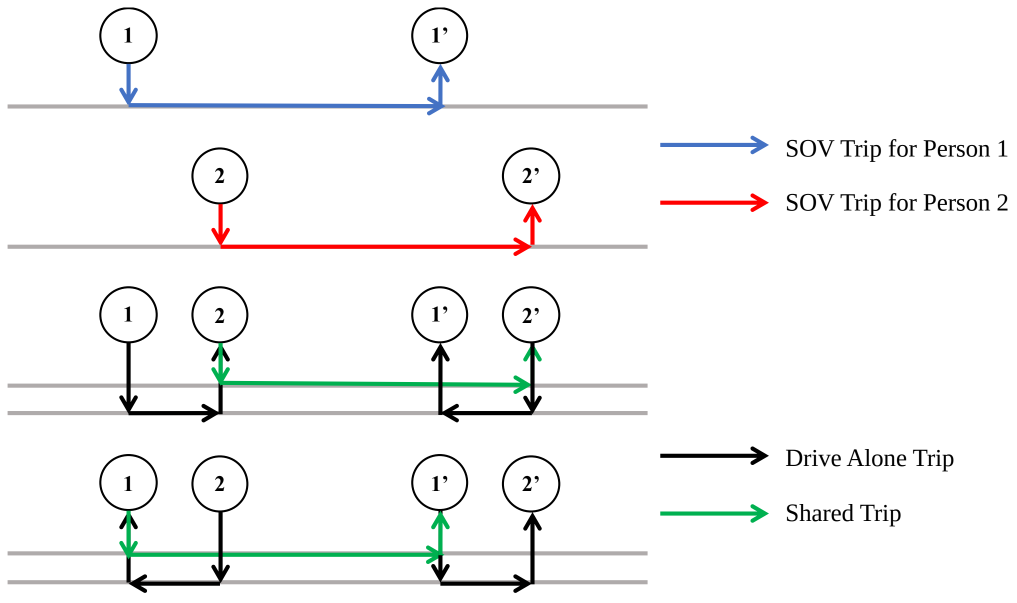


Figure 3. Three Basic Carpooling Assignments.

As shown in Figure 4 and Figure 5 below, triangular dots are party A's OD coordinates and rounded dots are the OD coordinates of the party B. The red dot represents the origin and blue dots represents the destination. A thick portion of the line shows the carpooled portion of the trip, while the black arrows show the driver's whole trip. As seen in Figure 5, even two trips start with the same coordinate, the two traveling schemes are very different in traveling paths, so driver and passenger roles are very important in carpool assignment.

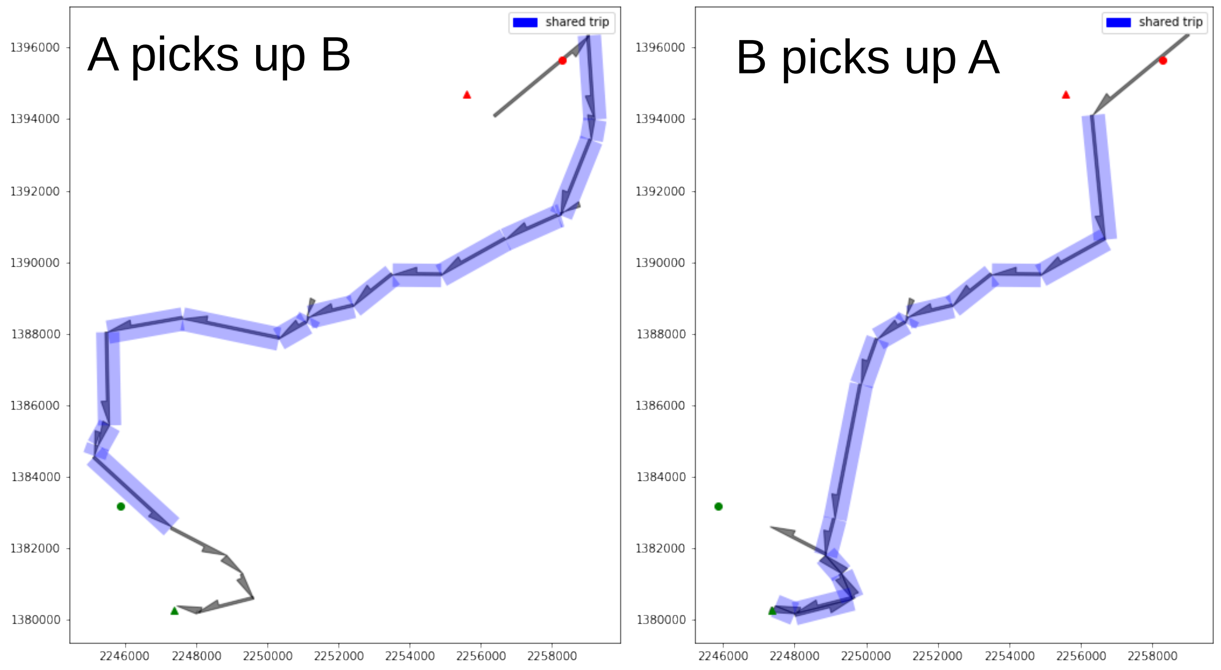


Figure 4. Visualization of the Carpool Paths Generated by CarpoolSim (Case 1)

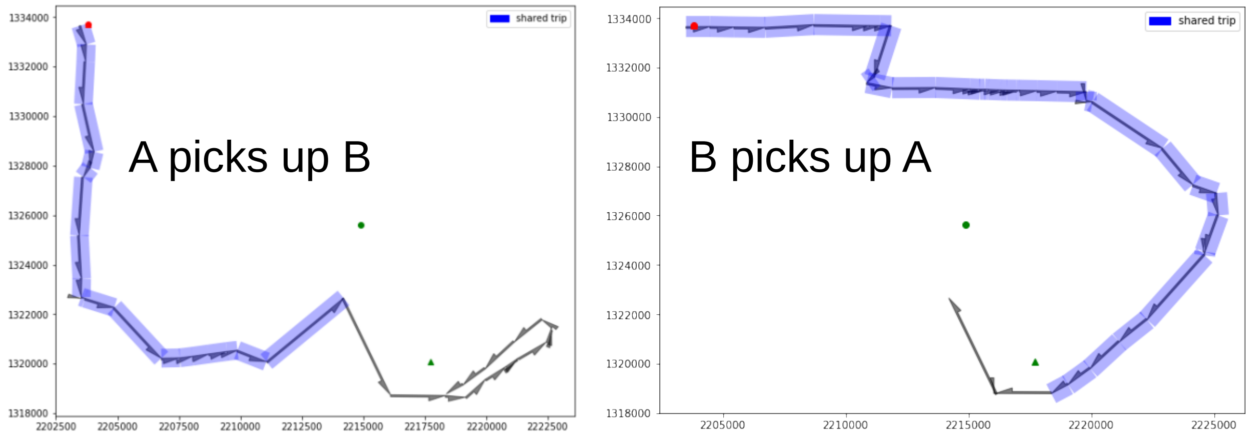


Figure 5. Visualization of the Carpool Paths Generated by CarpoolSim (Case 2)

3.3 Trip Spatiotemporal Clustering

The number of carpooling pairwise assignments grows as a polynomial function (i.e., $O(n^2)$) as the number of trips n grows. Hence, it is important to cluster trips into smaller groups to enhance computational efficiency. The idea is to only cluster trips sharing similar depart time and origin destination locations. Trips are first clustered based on location information. For each location-grouped cluster, trips are separately clustered based on temporal information (e.g., difference in departure time). Finally, the spatiotemporal clustering is formulated as the joint intersection of two separate schemes.

In practice, trips are first clustered by their origin TAZs. Within each group, trips are clustered using the DBSCAN method, by setting search radius parameters as a maximum tolerated departure time difference (e.g., 5-min), and the minimum neighboring number parameter n equals to two (two neighbors within cluster). In this way, we generate a number of clusters, and only trips within the same cluster can be considered for potentially carpooling together. In other words, if two trips are separated to two clusters, we assume that the trips are not likely to be carpooled together. In practice, many trips are clustered into an isolated cluster containing only the one trip in the cluster; hence, the optimization process is not needed for that trip. The second phase of this research will group using the origin TAZ, instead of using both the origin and destination TAZs, and the system will allow carpool pickups and carpool drop-offs at any TAZ along the path that does not add too much travel time to the original trip. The coding for this process is complex, and will be reported in future work and publications.

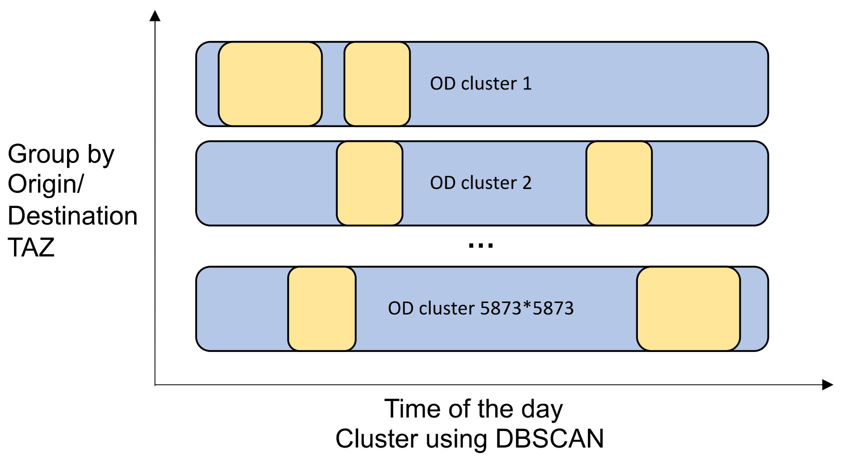


Figure 6. Demonstration of a basic trip clustering method

Location clustering over origin TAZs is reasonable considering we are discussing community-based carpooling, and a community usually contained in one single TAZ. However, the major disadvantage of this clustering method is the method ignore cases like a driver goes across the street and picks up a passenger in a neighboring TAZs. Future work will assess potential carpool joins across TAZs that fall within a set of time and distance tolerances. Geographic proximity doesn't always lead to efficient routing. In reality, one might need to reroute for more than a mile just to pick up a passenger in an adjacent TAZ. Additional shortest path routes will also need to be developed for local roads to support adjacent TAZ origin pairing. For the time being, this thesis only clusters trips by departure TAZs for good simplicity and interpretability.

After clustering trips into groups, the next step is to solve the carpool optimization problem within each trip cluster. A feasibility matrix is used to store measurements between proposed trip pairs. Two methods are developed, one empirical method and one formal integer linear programming method. They will be discussed in the next two sections.

3.4 Feasibility Matrix

Given a trip cluster with m travelers, there are $m \times (m-1)$ number of possible carpooling assignments considering all combinations with roles. Notice that there are also m original SOV travel plans. We can use a single feasibility matrix with size $m \times m$ to denote all these potential travel plans. This is feasibility matrix is used for each cluster. Only the trips within the same cluster can be considered as carpool trips. Trip pairs between cluster groups will never be considered to form a carpool.

Assume *F* denotes the feasibility matrix. Given m travelers in the system, the feasibility matrix is a $\langle x \cdot x \rangle$ square matrix, where *i* th row denotes the assignments where traveler *i* is the driver and column *j* denotes the assignments where traveler *j* is the passenger. The matrix entry *(i, j)* denotes a carpool relationship where the driver *i* carpooling with the passenger *j* and form a carpool trip. The diagonal entries *(i, i)* are used to denote the original drive alone trips. For any entry, let value 1 denote two trips are carpool-able and let 0 denote the opposite case. We can settle multiple filters to further find as many infeasible carpooling plans as possible. One simple condition is that only when the departure time difference is less than 15 minutes can the two SOV trips considered to form a carpool trip. Other filters include considering demographic characteristics, personal preferences, etc. For example, person A_i only wants the form a carpool driving his own vehicle. Then, we can set all entries along the *i* th column to zero (except its own diagonal value *(i, i)*), eliminating the assignments where the traveler forms a carpool as a passenger.

We can settle multiple filters to further exclude most ones in the of the $\langle x \cdot x \rangle$ feasibility matrix. Just like clustering trips into groups, we can shrink the search space of the problem and consider only a small number of potential suggested carpools.

Similar to the idea of feasibility matrix, we define different matrices to store important values, including total vehicular travel time, absolute difference of travel departure time, etc. Using those values, plus filtering criteria, we can filter out carpooling assignments unlikely to happen. The following filters are employed.

- The difference in original departure time is less than a threshold (e.g., five minutes)
- For the proposed carpool trip, the driver's extra cost (expanded reroute traveling time) is less than a threshold (e.g., 15 minutes)

After generating a feasibility matrix that considers multiple conditions, the matrix can be used as an input to both the bipartite method and linear programming method. Moreover, the matrices storing other values (e.g., vehicular travel time) can be queried any time. Those matrices and the feasibility matrix together are all called shareability matrix, which means they measure the utility/cost/possibility between proposed carpooling assignments with role.

3.5 Two Methods for Optimal Solutions

3.5.1 Bipartite Method

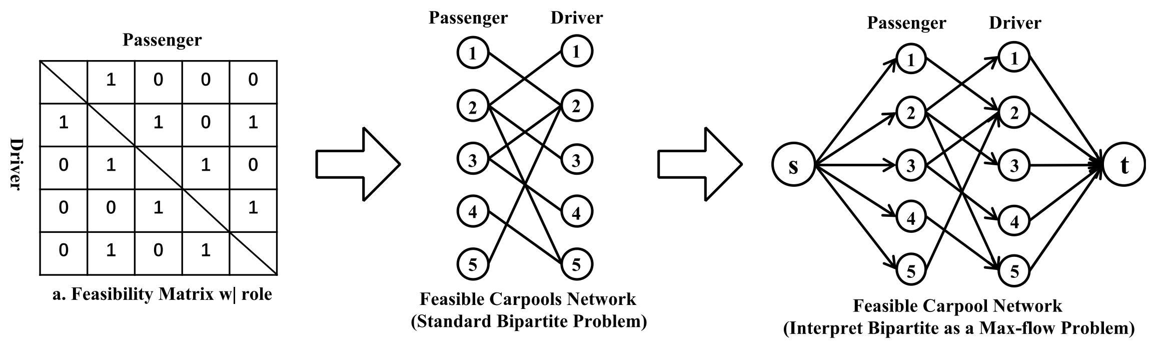
To solve the problem efficiently, the researcher proposed an empirical method borrowing bipartite algorithm plus heuristic decision rules to find the optimal solution (Figure 7). The bipartite method can be reduced to a maximum-flow problem. We define the problem's objective as: given the feasibility matrix, find the maximum number of traveling carpool pairs. The proposed algorithm is shown as follows:

Step 1. Assume each person can be both driver and passenger at the same time, and based on the feasibility matrix, generate a bipartite pairing problem to find the maximum number of trip pairs. Notice that the assumption (i.e., one can be both driver and passenger at the same time) is untrue, but the conflicts caused by this assumption will be solved later using heuristics.

Step 2. To satisfy the bipartite algorithm assumption, one person cannot play two roles at the same time in two separate carpool trips. When there is a conflict, the directed graph can either form a singly linked chain or a loop (i.e., singly linked chain with both ends connected), as shown in Figure 8 as an example. In other words, it cannot form any structure with one node having three connections to other nodes. In this way, we can greedily solve conflicts from the start of the chain and get the optimal assignments of finding the maximum number of carpooling pairs.

Bipartite method schematics are shown in Figure 7, Figure 8, and Figure 9. Figure 7 shows an example of a feasibility matrix for a carpool network (as a bipartite problem), and then convert this bipartite method to a Max-flow problem. The paths of the Max-flow problem solution can be used to construct a graph that connects travelers, as shown in the right plot in Figure 8.

By observing this Max-flow problem, it has two optimal solutions shown by Figure 8. One only contains chains and the other only contains loops. Figure 9 gives the heuristics rule to break up conflicts remember that one cannot become driver and passenger at the same time. Basically, the rule is to de-select links in turn until there is no conflicts. Since the graph structure is limited to singly linked chain or loop. The rule will guarantee reaching an optimal result in terms of keeping the most number of carpool trips.


Passenger
Driver
a. Feasibility Matrix w| role
Passenger
Driver
Feasible Carpools Network
(Standard Bipartite Problem)
Passenger
Driver
s
t
Feasible Carpool Network
(Interpret Bipartite as a Max-flow Problem)

	1	0	0	0
1		1	0	1
0	1		1	0
0	0	1		1
0	1	0	1	

Figure 7. Use Bipartite method to represent the feasibility matrix

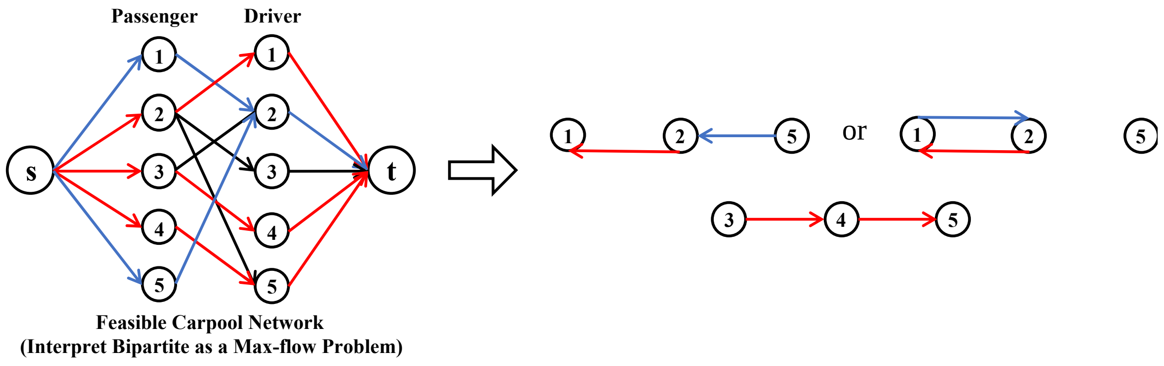


Figure 8. Two optimal outputs by bipartite algorithm

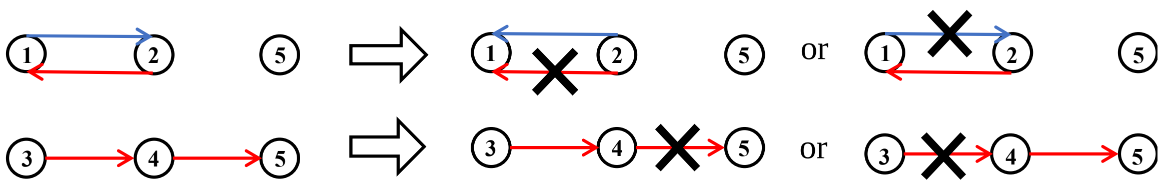


Figure 9. Heuristics to break up the role assignment conflicts

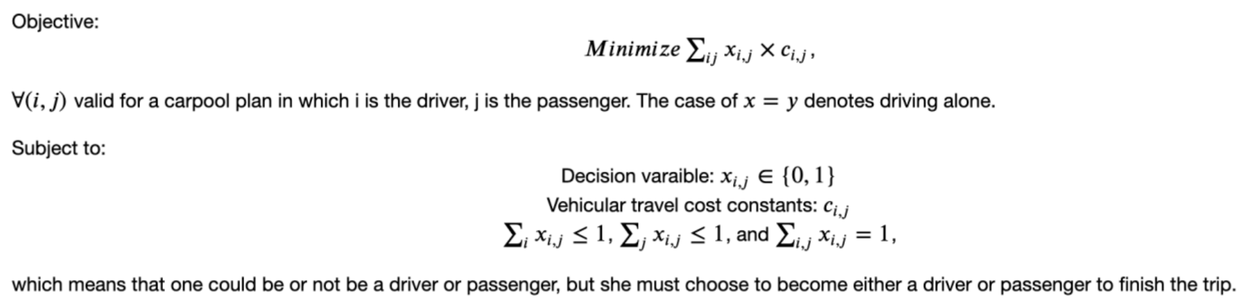
Proposition 2. In carpooling assignment problem, the bipartite algorithm only outputs singly linked chains or simple loops with both ends connected. One can index the links as [1, 2, …, n]. By preserve either odd or even indexed links and discard the other links, one can guarantee that one of the assignments finds the optimal solution in terms of pairing maximum number of travelers together.

Proposition 2 is easy to understand as the max-flow context where link costs all equal 1, any traveler node can at most be identified as a passenger once and as a driver once. Otherwise, the system would violate the max flow constraints of 1 between node source/target node and any traveler node.

3.5.2 Linear Programming Formulation

The bipartite algorithm only tries to find the maximum number of carpool trips. However, to further study this optimization problem, we need to set more flexible objective functions. For example, one might want to minimize total vehicle traveling time in the system. One might want to maximize total utility for the system. Moreover, the constraints can also be very flexible. For example, the user could constrain the utility costs of carpooling to be no more than 10 percent higher than for people driving alone. The bipartite algorithm, although efficient, only aims at finding the maximum number of carpools. Hence, a linear programming solution will provide more flexibility and can solve the problem using a linear programming solver.

For linear programming, the most basic objective function is to minimize the total vehicular travel time. A linear programming formulation is generated as follows:


Objective:

$$\boldsymbol{Minimize} \sum_{ij} x_{i,j} \times c_{i,j} \,,$$

$\forall(i, j)$ valid for a carpool plan in which i is the driver, j is the passenger. The case of $x = y$ denotes driving alone.

Subject to:

Decision varaible: $x_{i,j} \in \{0, 1\}$

Vehicular travel cost constants: $c_{i,j}$

$\sum_i x_{i,j} \leq 1$, $\sum_j x_{i,j} \leq 1$, and $\sum_{i,j} x_{i,j} = 1$,

which means that one could be or not be a driver or passenger, but she must choose to become either a driver or passenger to finish the trip.

Figure 10. Integer linear programming formulation and its interpretation

Both the bipartite algorithm and the linear programming algorithm will be used and tested in the next section. For same input data, the results generated by these two methods is compared in Chapter 4.

CHAPTER 4. EXPERIMENT

Two experiments are undertaken using trip outputs of Atlanta Regional Commission's activity-based model (ABM). The trip outputs are generated from the ABM15 model version of the ABM for the calendar year 2015 scenario. As described in the last chapter, not all trips are considered for carpooling. Instead, only single occupant vehicle commute trips towards designated employment centers are considered in this experiment. The "employment centers" are designated TAZs identified by the Atlanta Regional Commission (ARC, 2017). The name and location of these centers can be seen in Figure 11. These centers cover a lot of areas and employment centers in the Atlanta metropolis, which is interesting for transportation/city planners and engineers. Also, the limited number of high-demand destinations makes the research more interpretable and aligns with the community carpooling study topic. To demonstrate the efficiency of the algorithms, all of the computation steps are run on a Macintosh with 32-GB memory 2.4 GHz 8-Core Intel Core i9 processor.

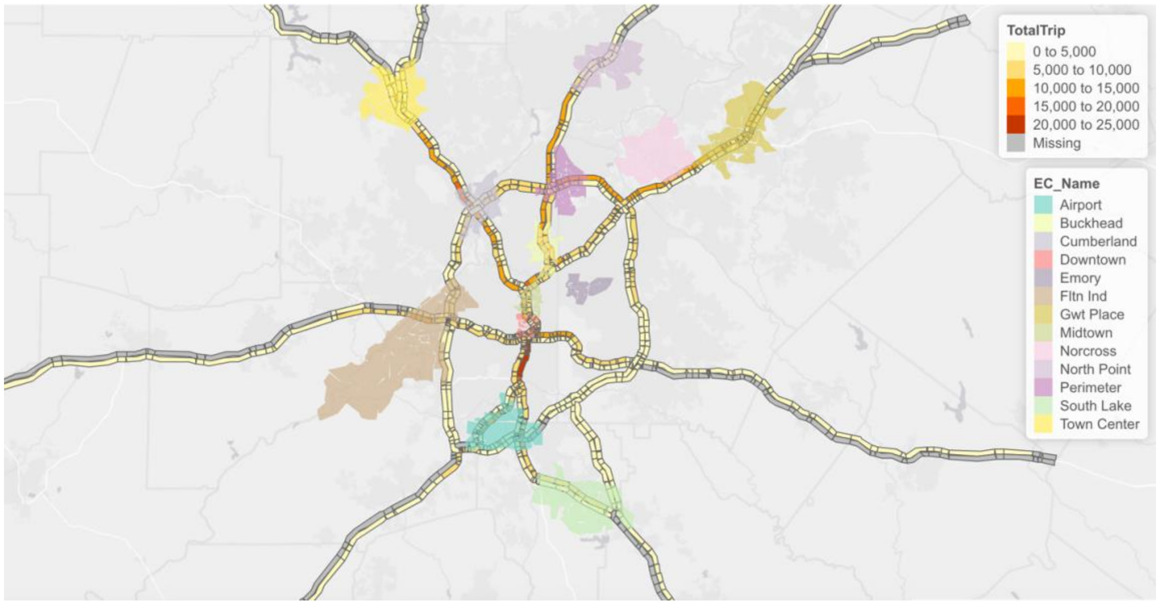


Figure 11 Employment Centers Identified by ARC

4.1 Experiment Settings

Only a small percentage of trips are fed into the analysis framework, given that there are many filter conditions to identify the trips of interest. For example, since carpool matches are only for SOV commuters driving from home to work, only those satisfying all of the above requirements serve as input to the analysis framework.

A description of filtering procedures this Georgia Tech research team used and the changes in data sizes are listed in the Table 2. More than 19 million vehicle trips are output by the ABM. Multiple filters are applied in series to remove trips that are outside of the project scope. Of the 19 million total vehicle trips, the number of trips with an employment center destination shrinks the trip total to 5,329,225 (27.7% of total trips). Among those trips, 2,293,265 (11.9% of total trips) are SOV trips. Among these SOV trips, 1,147,379 trips (5.96% of total trips) are for work (commuting trips). Among these work-related trips, the number of trips from work-to-home (instead of home to work) is 432,043 (2.3% of total trips). In summary, 432,043 trips started from home, ends at workplace, and the destination workplace is in a TAZ corresponds to employment center. Since we focus on inbound travel towards these designated employment centers, the number goes down to 385,880. Finally, the remaining 385,880 trips are used as the input of the analysis framework (Table 2).

Table 2. Counts and Percentage of Trips in Each Screening Step

Trip Conditions	Trips	Percent Retained from Previous Step	Overall Percent Retained
Total daily trips	19,235,737		100%
Trips involving ECs	5,329,225	27.7%	27.70%
SOV trips	2,293,265	43.0%	11.92%
Work-related trips	1,147,379	50.0%	5.96%
Home-to-work trip	432,043	37.7%	2.25%
Destination TAZ is an EC	385,880	89.3%	2.0%

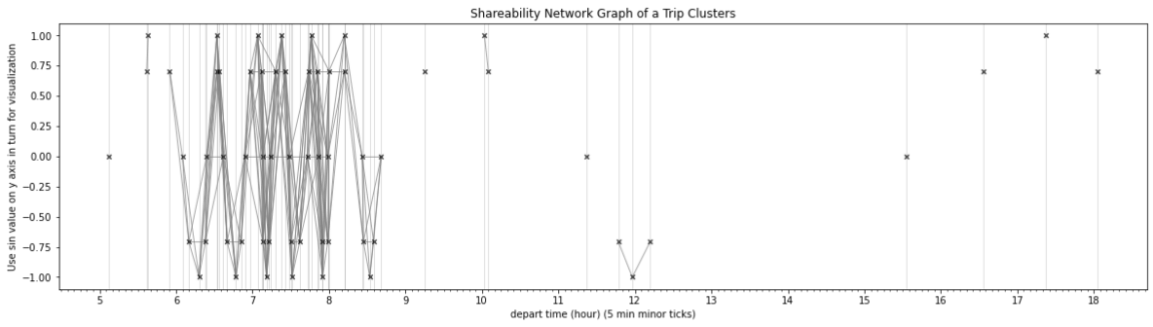
4.2 Disaggregate-level Results Analysis

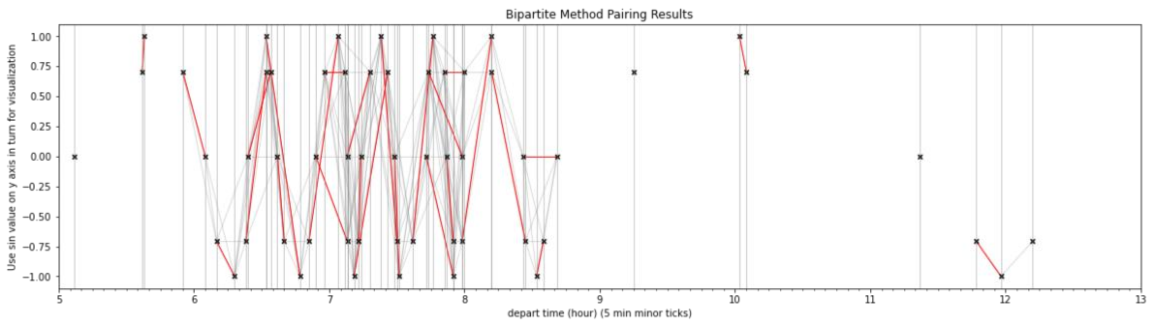
Using the same settings, the problem is run using both the bipartite algorithm (with heuristics) and the linear programming algorithm. To compare the performance of these two algorithms, all the steps before optimization use the same data set. Both algorithms are able to provide a solution with approximate aggregate numbers. One immediate question is to discuss the difference between the linear programming outputs and the bipartite method outputs. To compare the difference, one idea is to compare the quality of the paired data without aggregating data. In other words, compare statistical traits between the individual carpool plans.

One of the first things that can be done is to look into some trip clusters samples and visualize pairing results over the shareability network. Figure 12 shows the traveling cluster between traffic analysis zone (TAZ) #1557 and TAZ # 1583. Within the scope of interest, there are 60 trips between TAZ 1557 and TAZ 1583. As shown in the following graph, each node denotes a travel demand, and each link denotes a feasible carpooling plan.

The shareability network is constructed in a way that trip A can pick up trip B if trip A depart within 15 minutes ahead of the departure time of trip B. The x-axis of the graph denotes the trip's depart time while the y-axis is just jittered axis for visualization. In this example, a sinusoid function is used as input to the data index. The time difference of 15-minutes is used because the suggested carpooling system is not a real time dispatch application, but is rather assumed to be the results of negotiation between parties before the commute day. Notice the shareability network is such that even when a time difference is set to 5 minutes, many links still remain.

The bipartite picks the assignment plans either indexed as odd or even. By contrast, the linear programming algorithm chooses to make the assignments saving the most vehicular travel time.





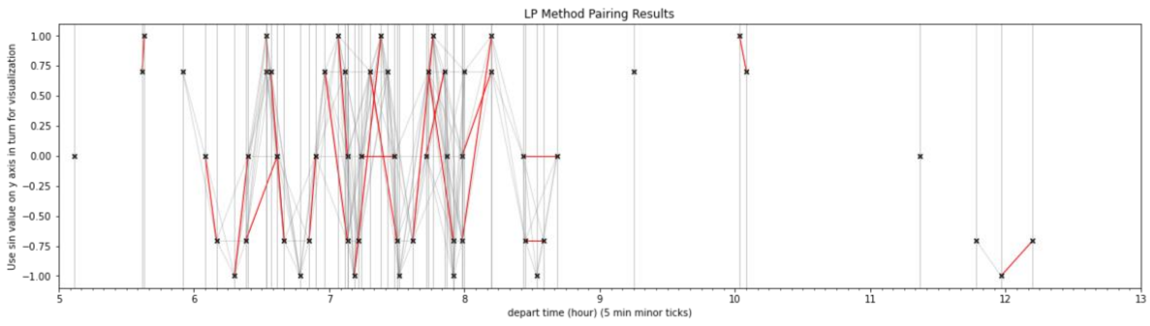


Figure 12 Shareability Network and the comparison of bipartite results and linear programming results.

By observation, only the travel changes for these carpool drivers need to be discussed. This is because the travel time for carpool passengers remains unchanged and only the drivers need to reroute to pick up and drop off passengers. For the purposes of the discussions that follow, all SOV case before assigning carpools will be identified as the "before" case and the scenario after running the carpool assignment will be defined as "after" case. In this way, two kinds of figures are plotted as shown in Figure 13: 1) before-after comparison of travel time for carpooling drivers; 2) reroute time for carpool drivers. The two graphs are logically related but emphasize on different contexts. The travel time plots focus on the overall performance of the carpooling system. The reroute time emphasizes the extra costs to the carpooling drivers. Notice that the before-case travel time histograms on travel time (i.e., the left, blue histograms) is not the exactly the same for different algorithms, because different algorithm may assign different SOV trips to form different carpools.

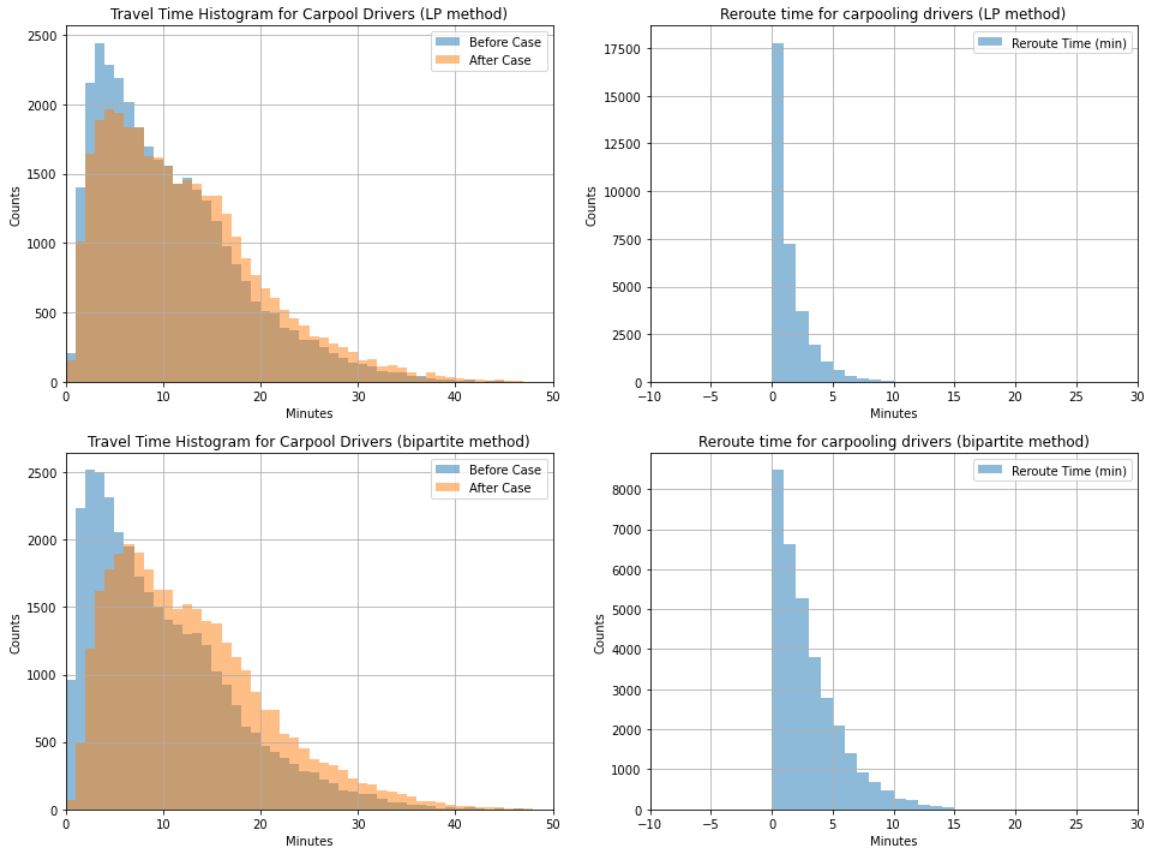


Figure 13. Change of traveling time among after-case carpooling drivers

While the overall distributions are not very different in shape, the reroute time distribution clearly shows that the linear programming method provides better solutions with much shorter reroute time (i.e., extra travel time), as the distribution tails are much shorter. This finding is *not* obvious as the linear programming method only guarantees the systemic minimum vehicular traveling time with no constraints on individual cases. Although the bipartite method is able to assign more carpools (as it is the objective function of the method), the efficiency of each assigned trip is worse than that of linear programming. The heuristics rules in bipartite-based algorithm fail to sort out the best solution when there is a conflict. Moreover, one high quality carpool pair may outweigh five low carpool pairs.

The inflation ratio for travel time is derived from the concept of inflation rate in economics, but the is difference in travel time instead of the price of goods. A graph showing the inflation rate versus original travel time is presented in Figure 14. This further confirms the findings that linear programming gives more efficient carpool assignments (i.e., higher quality). For each graph, the top and the right marginal plots correspond to the marginal distribution of data points along the dimension. A contour plot is estimated as a coarse estimate of the joint distribution. The main finding is that the mode (i.e., the highest peak of contour plot) of two methods doesn't completely overlaid, with LP's peak very close to zero inflation ratio while bipartite-based algorithm's peak is around 0.1-0.2. This further shows that the linear programming method is friendlier to carpool drivers.

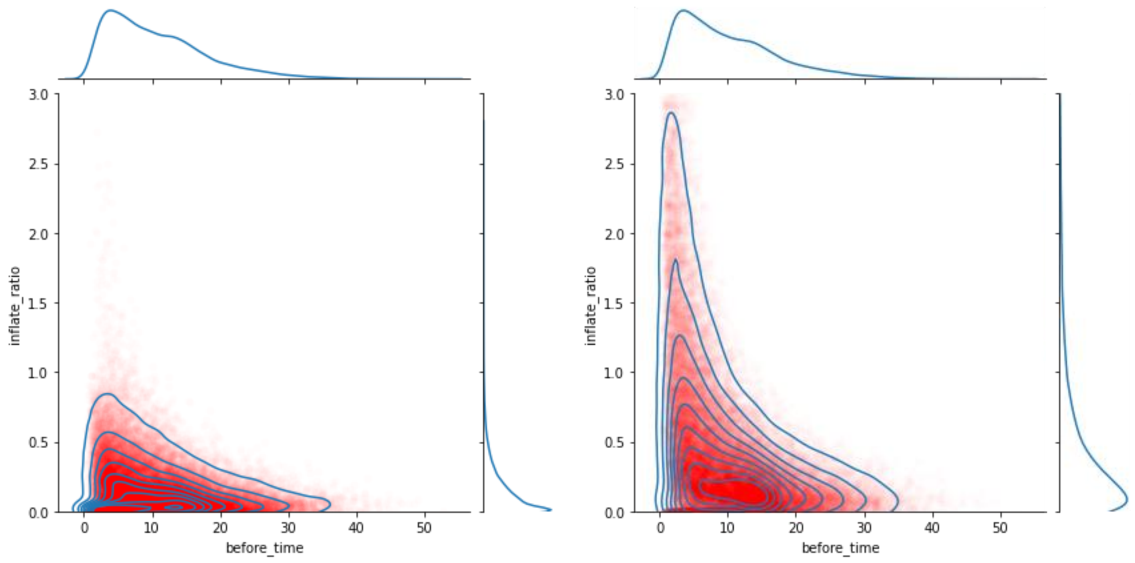


Figure 14. Inflate ratio vs. travel time among carpooling drivers (LP vs. Bipartite Method)

4.3 Aggregate-level Results Analysis

Figure 15 compares the paired data. The two diagrams show the data in linear scale, with the left diagram showing the counts for the linear programming method, and the right diagram showing the counts for the bipartite method. Each specific's x-axis shows whether the trip within the same TAZ number or not. The blue bar represents the paired carpool trips while the orange bar represents the remaining SOV trips. The bipartite algorithm finds more local pairs traveling within the same TAZ. The linear programming algorithm finds a smaller number of trips because the gain in mileage/gas savings for short range trips is very small, or even negative considering TAZ access costs. On the contrary, the bipartite method prefers to pair as many pairs as possible as long as the reroute cost is within a reasonable threshold.

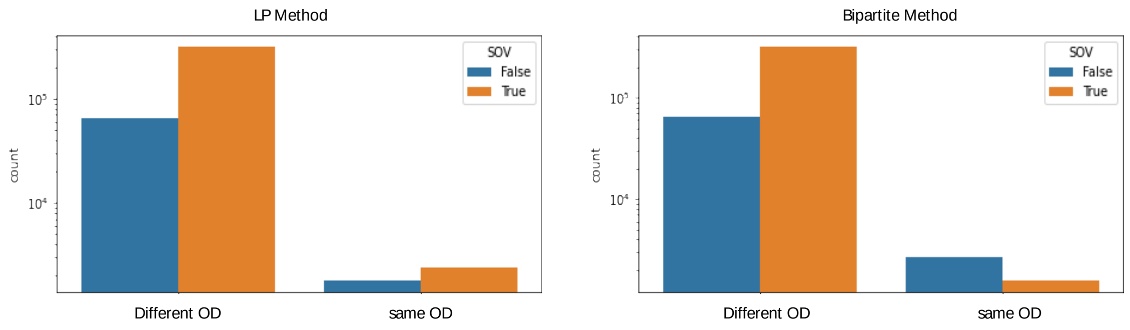


Figure 15. Comparison of Two Algorithm's Carpool Pairing Ratio for Trips between TAZs and within TAZs.

Instead of discussing trip details for individual trips, the overall impact on the system can be discussed over space and time. From last paragraph, one concern is that many carpool assignments are locally paired, which could be caused either by a biased towards short-range trips for the travel demand model, or by the optimization algorithms.

All carpool trips in Figure 15 are paired only when the OD pairs are the same. The plot for the linear programming algorithm showed that: 1) the short-range trips with same origin/destination TAZs only represent a very small portion of trips; 2) short range trips are less likely to be paired than long distance trips; However, the same plot for the bipartite algorithm shows that it is more likely to pair short range trips, assuming the a reroute travel time of five minutes is acceptable, which is not good for the carpooling system to deliver good quality assignments.

One simple method is to explore the trip's distribution is by aggregating trips by their destination TAZ or employment center, as shown in the bubble plot Figure 16. In Figure 16, the original number of SOV trips attracted and the percentage of carpooled results are plotted, with the size of the bubble corresponding to the number of carpool trips. This plot indicates that while Midtown attracts the greatest number of trips, only less than 10 percent of the trips are carpool-able. By contrast, although Emory has the second smallest number of trip attractions, it has the second largest number of paired trips. The Airport destination contributes the largest percentage as well as the number of carpool-able commute trips.

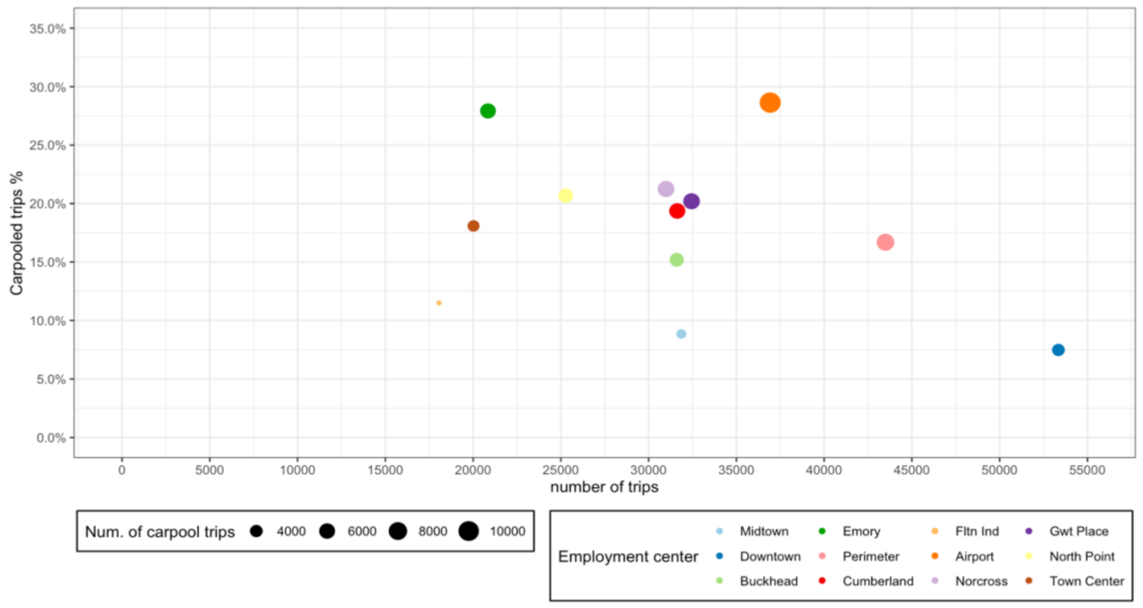


Figure 16. Bubble Plot for Each Employment Center Comparing SOV Trips and SOV Trips Assigned to Form Carpools

Figure 17 shows the counts of TAZs on the y-axis vs. the number of attracted trips for each TAZs. Each subplot represents the case for an employment center. This plot can show the heterogeneity of destination distribution over destination TAZs. Downtown and Midtown has lots of small TAZs that only attract a small number of trips given the assumption that destinations must be to the same TAZs. The airport and Emory have several outliers with very high number (over 5,000) of attracted trips, which is intuitive given their high demand.

Figure 18 shows bubble plots for carpooled percentage vs. total number of SOV trips (before consider carpooling). Each bubble corresponds to the trips to a destination TAZ. As the number of trips grows, a destination is more likely to pair more of its trips. This is expected because the experiment only considers pairing trips within the same TAZ.

There are three big outliers in the bubble plot, one is a TAZ in Emory and two others are TAZs in Airport. This is reasonable because the Emory University campus and hospital and airport terminals are large employment attractors.

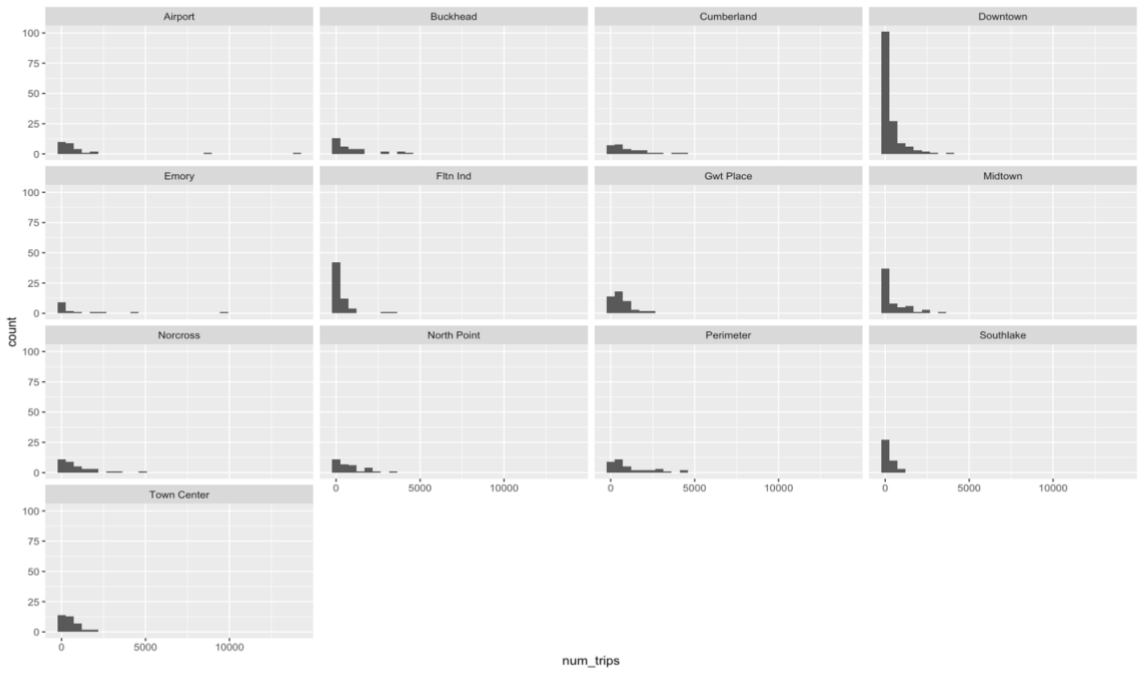


Figure 17. The Counts of TAZs vs. the Number of Attracted SOV Trips Grouped by Employment Centers

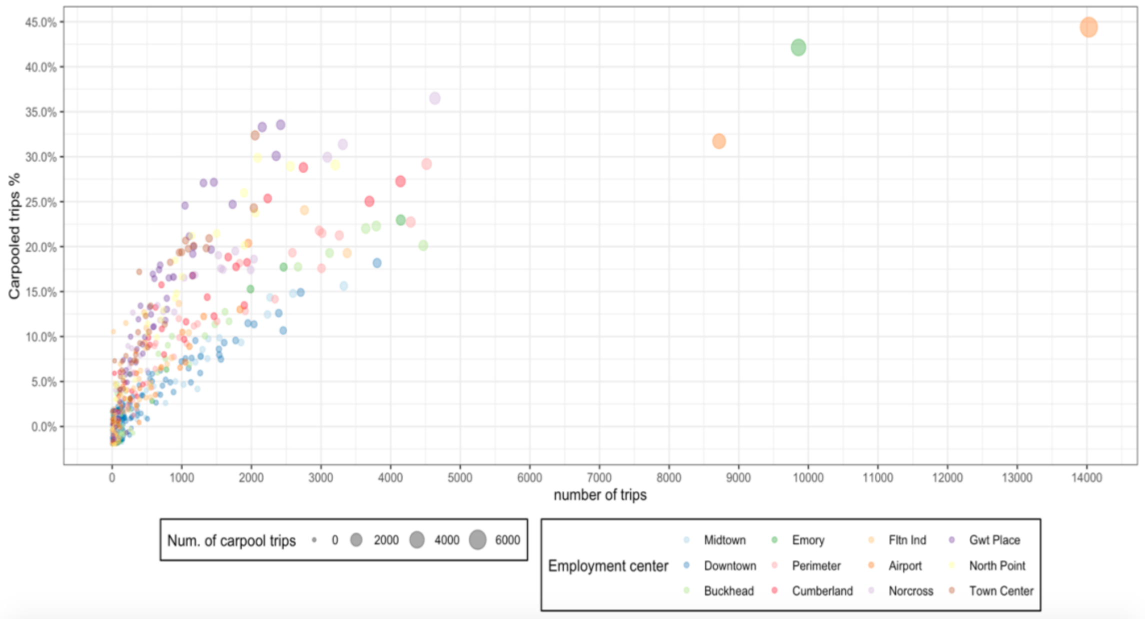


Figure 18. Bubble plot for each destination TAZ comparing SOV trips and SOV trips assigned to form carpools

4.4 Network-level Results

The spatial distribution of trips is also an important topic. One interesting method is to analyze the carpooling assignment's distribution on traffic networks and thus its impact on alleviating the pressure on the metropolitan's skeleton highway system.

As shown in Figure 19, we plot the absolute traffic volume distribution in the before scenario. Not surprisingly, most trips are inbound traffic, and many outbound links are not even used, as they are plotted in grey color and marked with a "Missing" legend. The Northern part of the city (i.e., I-85 inbound, I-75 inbound, US-19 inbound, and the northeastern corner of I-285 towards the west) has a large volume of loaded traffic (over 10,000 trips). A small northbound segment towards the city's downtown has over 20,000 trips loaded.

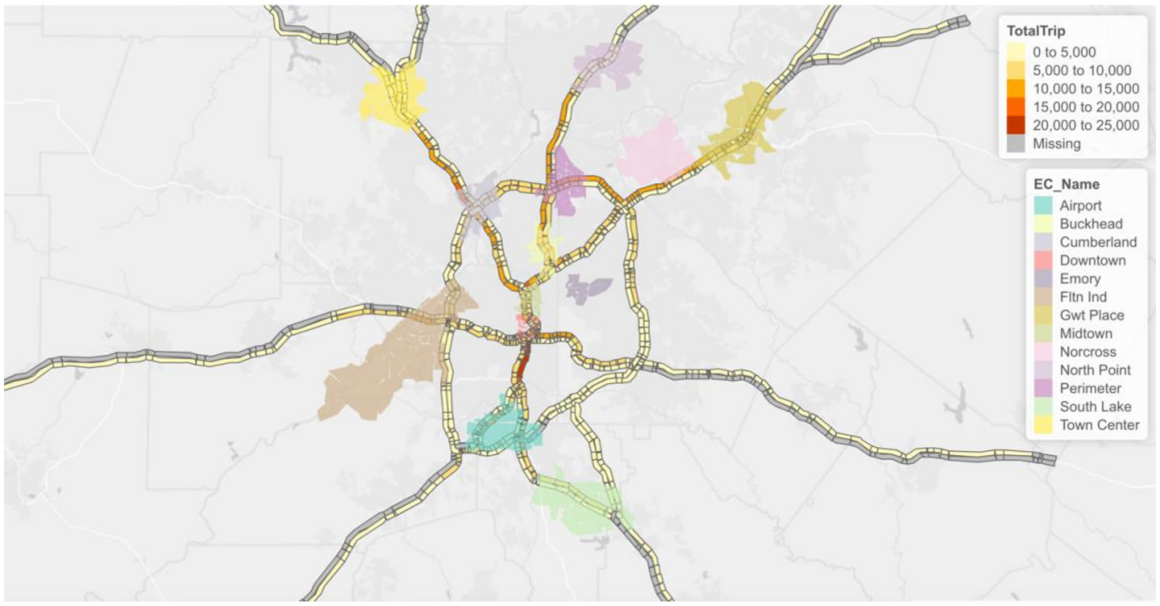


Figure 19. Traffic volumes for all before-case SOV commute trips along skeleton expressway network

Among these SOV trips, the distribution of the before case SOV trips paired in after case shows a slightly different pattern, as shown in Figure 20. US19 inbound traffic has a largest carpool travel counts, and it is higher than that of I-75 and I-85 inbound. The reasons of that are not clear and we need more analysis. It could because of the origins of I-75 and I-85 travel are sparser, making the possibility of trips being paired together lowered (since the algorithm is based on origin locations). Other employment or demographic reasons might be in play for the SOV trips happening in the northern region of Atlanta. For example, people living in Buckhead (north suburbs) have a higher income and may be more likely to drive alone, providing more SOV trips that could form carpools.

Figure 20 plots the volume counts of carpooling trips over the skeleton network. The pattern shows that most carpool trips happen for the inbound traffic along US-19 near I-285 in the north of Atlanta with more than 2,000 carpool trips. There are over 1,500 carpool trips in the southern region of downtown Atlanta along I-75/I-85 north-bound.

While the inbound traffic from the north along I-75, I-85, and US-19 is similar (from 10,000 to 15,000 carpool trips), there are more carpooled on in US-19 (Figure 19). This shows that SOV trips along US-19 have more potential to form carpools.

Figure 21 plots the percentage of assigned SOV trips over the skeleton network. The pattern of this distribution is different from the other two diagrams. It shows that inbound traffic along I-285 has the highest ratio of carpooled together (over 30 percent). This reveals that trips toward the Airport region (i.e., tiles in green) have the highest potential for being paired. Consider the TAZ of airport terminal is attracts lots of traffic, it is normal that lots of SOV trips have this specific TAZ corresponds to the same airport terminal, making the trips easier to form carpools. This will be further supported when the next algorithm for the system is implemented that allows TAZ drop-offs at any TAZ along the commute path and to adjacent TAZ destinations (not included in this report, but is being released in the Phase 2 activity for this work).

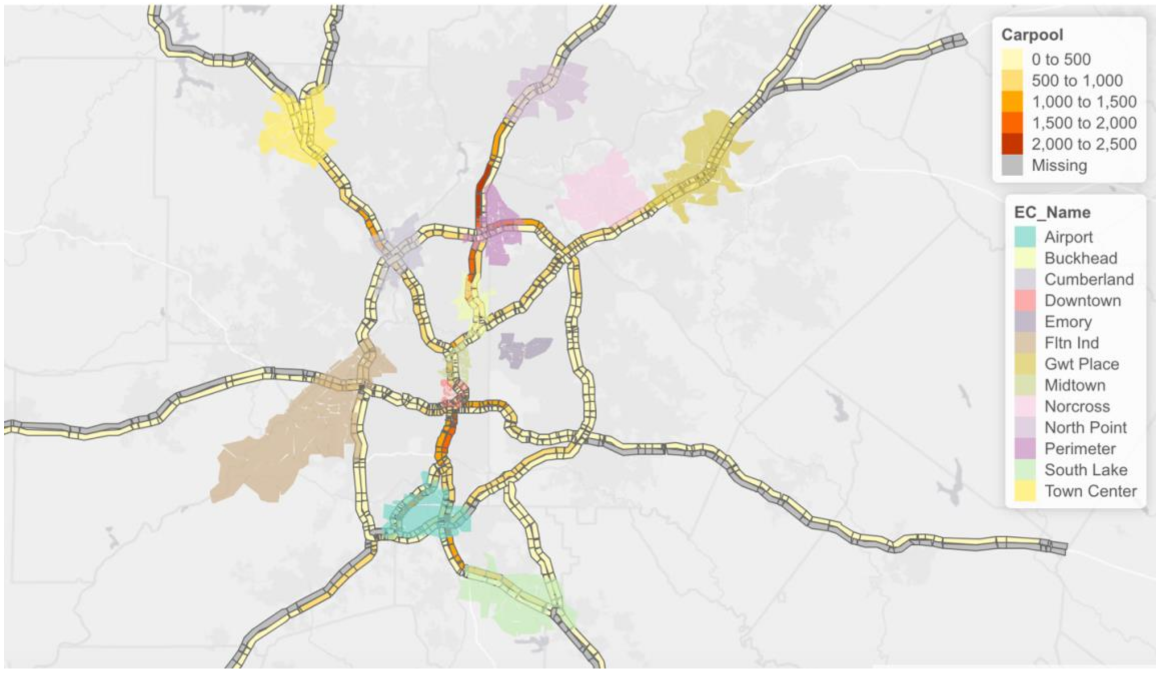


Figure 20. Traffic Volume distribution of carpooled SOV commute trips along the skeleton expressway network

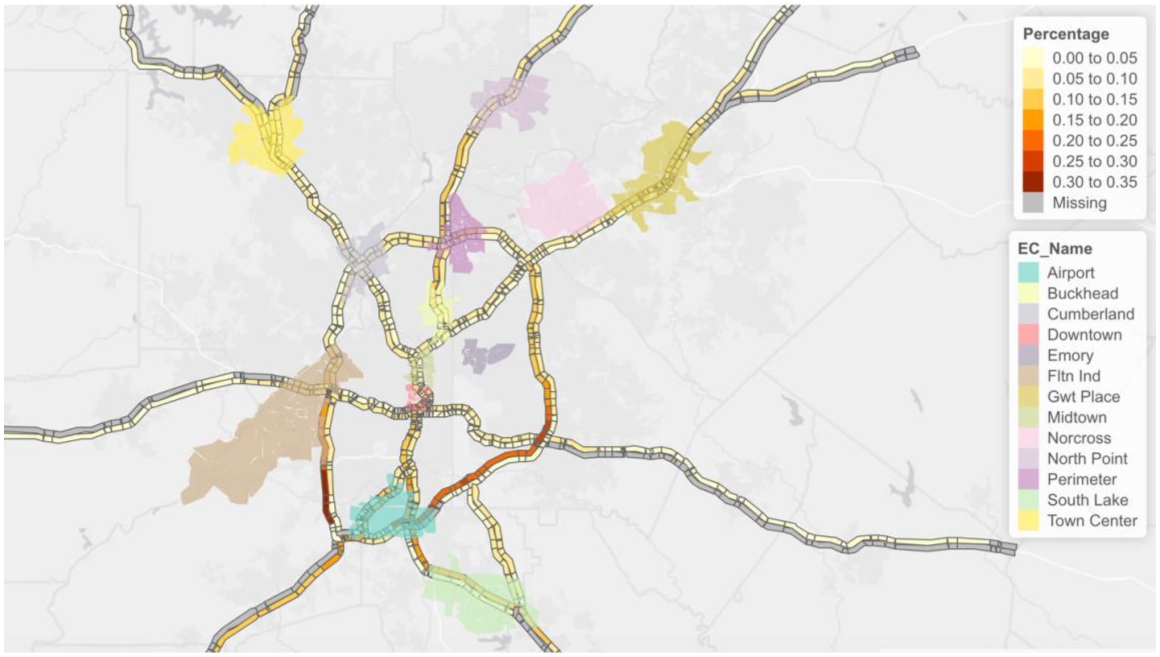


Figure 21. Percentage distribution of carpooled SOV trips along the skeleton expressway network

In conclusion, the experiment results show that community-based carpooling is useful in decreasing the traffic burden during peak hours. The analysis results can be helpful for planning activities like finding suitable places for settle up HOT lanes. Moreover, evaluating environmental beneficial impacts are also easy starting from the outputs.

The framework's is suitable not only for ARC's ABM but also for all CT-Ramp activity-based model's trip outputs. Generally speaking, the framework can analyze all individual-wise travel results with only a number of modifications based on different research goals. For example, by modifying algorithm, transit agency can use the model to find the optimal community-based vanpool schedule.

CONCLUSIONS

This thesis has developed an analysis framework to estimate the upper-bound for community-based carpooling trips among SOV travelers, given a set of reasonable spatiotemporal constraints. A set of experiments are run based on the trip outputs of the activity-based travel demand model employed by ARC. The results are promising that even under strict constraints, the model finds that around 13.5% of SOV commute trips in the morning peak period, from home to major employment centers, sharing the same departure TAZ departure and arrival TAZ, and sharing a similar departure time window could be paired to form carpools. The model outputs are further analyzed over space, time, and travel network. Two different optimization algorithms are proposed, one quickly finds the optimal solution in terms of minimizing the number of carpooling pairs while the other finds the systemic optimal value in terms of minimizing total vehicular hours. Analysis shows that the bipartite-based algorithm finds the greatest number of carpools, but fails to generate high-quality pairs for short range trips. The integer linear programming solutions generates highest-quality carpool pairs designed to reduce total VMT and therefore energy use and emissions.

A second phase of research has been purposed that will extend the analysis framework in the following aspects: 1) analyze vanpool/transit operations in smaller areas; 2) implement a simulation platform to simulate stochastic popup travel demands thus to simulate different assignment algorithms and evaluate its performances; 3) improve clustering methods in the framework and improve algorithm.

REFERENCES

1. Davidson, W., Vovsha, P., Freedman, J. and Donnelly, R., 2010, September. CT-RAMP family of activity-based models. In Proceedings of the 33rd Australasian Transport Research Forum (ATRF) (Vol. 29). Website link: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.455.2553&rep=rep1&type=pdf

2. Zhao, Y., M.O. Rodgers, and R. Guensler (submitted). Comparing Predicted Travel Behavior of Demographic Groups using Activity-based Models with Path Retention (19-03916). 99th Annual Meeting of the Transportation Research Board. Proceedings. Washington, DC. January 2020.

3. Calvo RW, de Luigi F, Haastrup P, Maniezzo V. A distributed geographic information system for the daily car pooling problem. Computers & Operations Research. 2004 Nov 1;31(13):2263-78.

4. Cruz, M.O., Macedo, H. and Guimaraes, A., 2015, November. Grouping similar trajectories for carpooling purposes. In 2015 Brazilian Conference on Intelligent Systems (BRACIS) (pp. 234-239). IEEE. doi: 10.1109/BRACIS.2015.36.

5. Jamal, J., Montemanni, R., Huber, D., Derboni, M. and Rizzoli, A.E., 2017. A multi-modal and multi-objective journey planner for integrating carpooling and public transport. Journal of Traffic and Logistics Engineering Vol, 5(2). doi: 10.18178/jtle.5.2.68-72.

6. Santi P, Resta G, Szell M, Sobolevsky S, Strogatz SH, Ratti C. Quantifying the benefits of vehicle pooling with shareability networks. Proceedings of the National Academy of Sciences. 2014 Sep 16;111(37):13290-4.

7. Zhang, H. and Zhao, J., 2018. Mobility sharing as a preference matching problem. IEEE Transactions on Intelligent Transportation Systems. doi: 10.1109/TITS.2018.2868366.

8. Qi, X., Wang, L. and Wang, X., 2016, July. Optimization of carpooling based on complete subgraphs. In 2016 35th Chinese Control Conference (CCC) (pp. 9294-9299). IEEE.

9. Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E. and Rus, D., 2017. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment.

Proceedings of the National Academy of Sciences, 114(3), pp.462-467. doi: 10.1073/pnas.1611675114.

10. Hsieh, F.S. and Zhan, F.M., 2018, July. A Discrete Differential Evolution Algorithm for Carpooling. In 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)(Vol. 1, pp. 577-582). IEEE. doi: 10.1109/COMPSAC.2018.00088.

11. ARC, Atlanta Regional Commission (2017).

https://cdn.atlantaregional.org/wp-content/uploads/5-regionalsnapshot-employmentcenters-oct2017-web.pdf

APPENDIX A. PROBLEM DEFINITION & NOTATION

Assumed all trips of interest are denoted as a set $G = \{g_1, g_2, ..., g_n\}$, where n is the total number of interested trips to be analyzed. A specific $g_i \in G$ stands for a specific trip demand. A trip demand may contain a party of one or several people sharing completely same trip demand. In this case study, the trip demand always corresponds to a drive-alone commuter. For any trip demand g_i, it contains the following spatiotemporal information ho_i, d_i, l_i, a_ii, which corresponds to the origin, destination, departure (leaving) time and arrival time of the trip demand g_i.

Two parameters, Δ and Γ, controls the lower-bound of carpooling quality and used as controlling thresholds to filter infeasible trips. $\Delta \geq \max_i \{\delta_d, \delta_p\}, \forall g_i$ stands for the maximal tolerable modification in departure time or pick up time. In the carpooling settings, either the passenger needs to meet driver at pick up location δ_d minutes earlier, or the driver needs to depart δ_p minutes earlier to get to the pickup location in time. This kind of schedule modification is a possible because the pairing plan is assigned before the traveling day. The analyses assume that setting Δ to between 5 and 15 minutes is reasonable as this will not significantly change daily commute routine. Otherwise, the two individual trips do not form a good quality carpooling trip. Notice that in the settings $\delta_d \times \delta_p$ 6= 0, meaning that δ_p and δ_d cannot be zero at the same time. In other words, either the driver depart earlier or the passenger meet the driver at the pick up point earlier. When Δ is a small number (e.g., 30 seconds), given the system's situation at a given time, the problem reduces to a ride-sharing optimization problem given the spatiotemporal ride-sharing contexts.

The other parameter Γ, denotes the maximal extra reroute time for the driver compared to his or her driving-alone case. Since the driver's reroute time is always larger than the passenger, only trips with the driver's reroute time smaller than a threshold Γ are considered feasible carpools. Γ controls the maximal extra tolerable reroute time for the driver.

For the analyses, k denotes the maximal number of trip demands served in a ride-sharing vehicle. When $k \geq 3$, the optimization problem becomes hard to solve. Also, previous studies found that $k \geq 3$ doesn't contribute too much to system performance. In this paper, we only discuss the case of $k = 2$. To be more specific, in the carpooling settings, a driver either drives alone or carpool with another commuter during the trip.

APPENDIX B. TWO CARPOOLING PLANS

A carpooling assignment means assigning multiple trips each of which uses one vehicle to form two-person carpools. In carpool formulation, unlike the taxicab service or ride hailing, the vehicle belongs to the commuter/owner. Because the system only considers carpooling among two travel demands ($k = 2$), there are only two carpool scenarios between trip i and j, that is either i acts as the driver and j acts as the passenger or vice versa. In the carpooling contexts, assuming i and j forms a carpool trip, the trip sequence can only be either $[o_i \rightarrow o_j \rightarrow d_j \rightarrow d_i]$ or $[o_j \rightarrow o_i \rightarrow d_i \rightarrow d_j]$.

B.1 Trip Filtering

Before enumerating all feasible carpooling combinations, the number of trip combinations needs to be narrowed down, filtering isolated trips and clustering trips into small groups based on their departure/arrival time and location. Based on the problem settings and research need, different clustering plans can be devised. The key is to filter out infeasible carpooling combinations, but not all infeasible combinations.

B.1.1 Trip Filtering with Approximate Destinations Constraints (Naive Scenario)

Simply grouping trips by their origin and destination serves as a starting point. Because the study assumes a community-based strategy, carpools are only formed when they start in the same origin transportation analysis zone. Moreover, these trips must drop off passengers at the same location (transportation analysis zone) in this simplified case. In this way, OD pairs can be considered separately, and carpooling assignment can be made in parallel.

Within each OD pair, run DBSCAN(*T;*Δ*'*,*n* = *2*) over the departure time. Δ*'* is the maximum difference in departure time between two original SOV trips. Δ' = $|l_d - l_p| \approx \Delta$ travel time for the driver picks up the passenger is small in community-based carpooling (Figure 21).

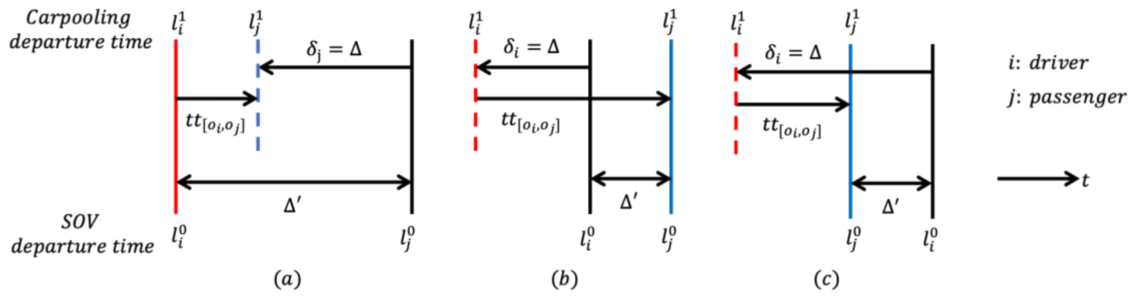


Figure 22 In pre-planned Community-based carpooling problem, Δ' is a good approximation for filtering trips when pickup time is relatively small

In Figure 22, assume i denotes the driver and j denotes the passenger. Case (a) represents the case that the driver i picks up passenger j at the same depart time, whereas the passenger needs to meet earlier than the SOV schedule. Case (b) and (c) stands for the driver i leaves earlier than the original SOV case to meet with the passenger j. The terms l_i^0 and l_j^0 stand for the original SOV depart time for driver and passenger respectively. Similarly, l_i^1 and l_j^1 denotes the carpooling depart time for driver and passenger respectively. There are only three possible cases considering i needs to pick up j. Notice that when $tt[O_i,O_j]$ is small compared to Δ , and there is $\Delta' \approx \Delta$. In this way, the filtering step on the original SOV departure time using DBSCAN(T;Δ',n = 2) is reasonable.

A large value (e.g., $\Delta'= 15$ min) instead of small number $\Delta = 30$ second is used because the carpooling assignment is not a real time dispatch but pre-planned ahead of the day of travel. In this way, a travel group can be further split into smaller groups. We use the DBSCAN to quickly sort out the clusters plus the potential feasible trips. In summary, the naive case can very quickly narrow down the searching scope in the expense of not considering integrating carpools with the case of dropping of passengers along the driver's travel path.

B.1.2 Trip Filtering without Approximate Destinations Constraints (Complete Scenario)

Similar to the naive case, trip pairing is performed in the complete scenario is limited to the same departure TAZ to reflect the community-based strategy. However, in this case a driver can drop off passengers along its travel if they are not sacrificing too much of their commute time. In the complete scenario, the destinations among carpoolers can be to different TAZs.

To query distance relationships between TAZs, a map of adjacent TAZs is established. For any trip g_i, given the shortest path between its origin & destination, all of the TAZs a trip passes are denoted as a sequence $Z^i = [Z_1,...,Z_k]$. Within this sequence of TAZs, if there is a feasible trip with OD among the sequence of TAZs, then a carpooling trip is considered. Interestingly, the DBSCAN can still be applied to cluster trips since they still share similar origins. Before applying DBSCAN, we need to cluster trips by destination along the shortest traveling paths. DBSCAN is used as the last filtering step to maximize its impacts in clustering trips into smaller groups.

APPENDIX C. GENERATE PRECISE SHARABILITY NETWORK WITH ROLE

The last section only filters out the isolated trips and cluster carpool trips into groups. In the settings, carpooling trips can only happen within the same travel group but not between different travel groups. To find the optimal carpooling assignment, a shareability network is constructed for each group. A feasibility matrix is then used to store the information for the shareability network.

A feasibility matrix $F = [f_{i,j}], \forall i,j \in \{1,...,n\}$ can be constructed to store all feasible carpool assignments among this trip cluster. For any $f_{i,j}$, a trip can be considered as a carpool if and only if:

$$|l_i - l_j| \leq \Delta,\ c_{i,j} = [tt(O_i,O_j) + tt(O_j,D_j) + tt(D_j,D_i)] - tt(O_i,D_i) \leq \Gamma$$

In the above formula, $c_{i,j}$ is the extra delay for the driver i. Only trips sharing similar departure time and short reroute time can be considered as a feasible carpooling pair.

APPENDIX D. FORMULATE CARPOOLING OPTIMIZATION PROBLEM

Let $x_{i,j}$ be the decision variable where 1 denotes the assignment of a carpool pair between driver i and passenger j. If $x_{i,j} = 1$, then there must be $f_{i,j} = 1$. Moreover, if a person is assigned to a carpool role as driver, then this person cannot be driver or passenger in the other groups. The optimization problem can be formulated as the following:

$$\text{minimize } \mathrm{P}_{i,j}\, x_{i,j} \times c_{i,j}$$

$$\text{subject to: } \mathrm{P}_{i}\, x_{ij} \leq 1$$

$$\mathrm{P}_{j}\, x_{ij} \leq 1 \quad \text{(D.1)}$$

$$\mathrm{P}\, x_{ij} + \mathrm{P}_{i}\, x_{ij} = 1$$

$$X_{ij}, j \in \{0,1\}$$

$$\{x_{ij}\} = \{f_{ij} | f_{ij} = 1\}$$

This integer linear programming (ILP) problem can be solved using any linear programming solver. We can start with a heuristics solution as a starting point the speed up the computation.

APPENDIX E. PROPOSITION 1

Proposition 1. Let's call the cyclic links circulating a TAZ the TAZ's minimum circulating loop (MCL). Assume the shortest path between TAZ's MCL is contained in the shortest path between two TAZ centroids, then the two-step procedure gives the sub-optimal paths between two coordinates bounded by a constant.

Proof

As one can see in Figure 23, almost every TAZ centroid is surrounded by major arterial roads (lines in purple), where the green links are the connectors from TAZ centroids to nearest one or few neighboring networks. In this way, the concept of minimum circulating loop holds for ABM network.

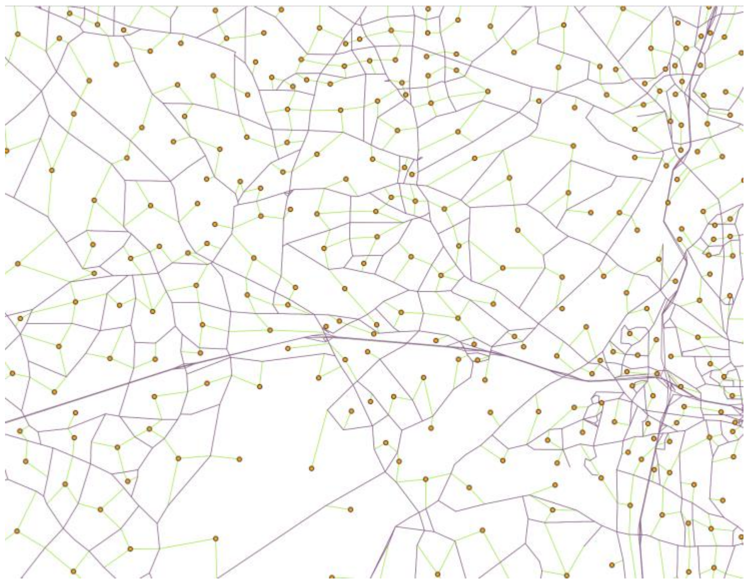


Figure 23 A Zoom-in part of ABM Network with TAZ centroids and its connectors

As shown in Figure 24, consider the shortest path from TAZ centroid A and TAZ centroid B. The shortest paths contain three segments: two connector links connecting centroid to network nodes, and the shortest paths between the two network nodes connects to the centroids. The latter shortest connecting path segment is also a very good candidate of the shortest paths between the MCLs of TAZ A and TAZ B. Assume this is true, then during stage two computations, we can just route the trip from their coordinates to the nearest node then to the two end nodes of the shortest path between two MCLs. In this way, we reconstruct a good approximation of shortest path, which is sub-optimal except adding some reroute time no worse than the perimeters of the two MCLs.

This proof may not be ideal, but it gives an upper-bound of the estimation's quality compared to optimality. In practice, it performs very well and achieves high computation speed. A manually constructed worse case is shown as follows. The estimated path is not the best option by observing that taking the link connecting TAZs on top is a better choice. However, this is rare to happen in the real case considering geometries of the network.

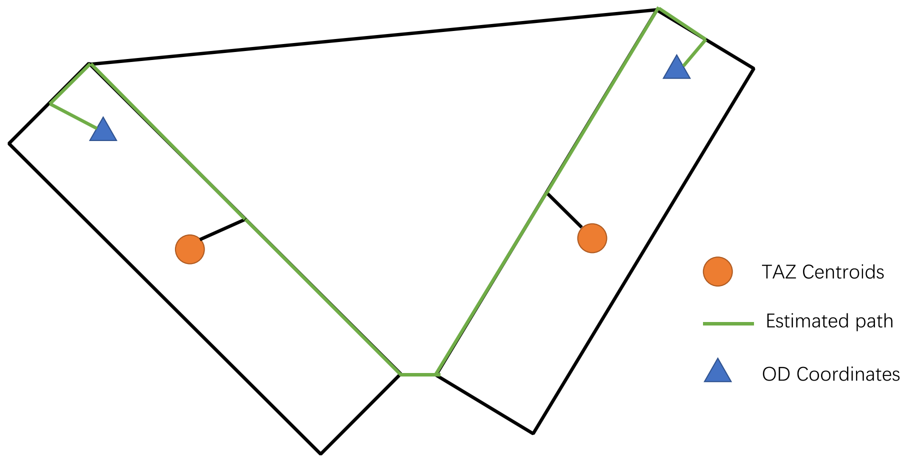


Figure 24 A Counter Example of Finding Non-optimal Shortest Paths