The Ridesharing Routing Problem with Flexible Pickup and Drop-off Points

November 2022

A Research Report from the National Center for Sustainable Transportation

Maged Dessouky, University of Southern California

Zuhayer Mahtab, University of Southern California

National Center for Sustainable Transportation

METRANS Transportation Consortium
USC | CSULB

TECHNICAL REPORT DOCUMENTATION PAGE

1. Report No. NCST-USC-RR-22-41	**2. Government Accession No.** N/A	**3. Recipient's Catalog No.** N/A
4. Title and Subtitle The Ridesharing Routing Problem with Flexible Pickup and Drop-off Points		**5. Report Date** November 2022 **6. Performing Organization Code** N/A
7. Author(s) Maged Dessouky, PhD, https://orcid.org/0000-0002-9630-6201 Zuhayer Mahtab, https://orcid.org/0000-0003-4793-2705		**8. Performing Organization Report No.** N/A
9. Performing Organization Name and Address University of Southern California METRANS Transportation Consortium University Park Campus, VKC 367 MC:0626 Los Angeles, California 90089-0626		**10. Work Unit No.** N/A **11. Contract or Grant No.** USDOT Grant 69A3551747114
12. Sponsoring Agency Name and Address U.S. Department of Transportation Office of the Assistant Secretary for Research and Technology 1200 New Jersey Avenue, SE, Washington, DC 20590		**13. Type of Report and Period Covered** Final Research Report (August 2021 – August 2022) **14. Sponsoring Agency Code** USDOT OST-R

15. Supplementary Notes
DOI: https://doi.org/10.7922/G2M32T3Z
Dataset DOI: https://doi.org/10.7910/DVN/3IWXBJ

16. Abstract
In major metropolitan areas such as Los Angeles County, ride-sharing systems can help reduce traffic congestion and increase the efficiency of the transportation system. This research project proposes three different solution approaches for solving the ride share routing problem with flexible pickup and drop-off points. The first is a dynamic programming-based route enumeration procedure that can be used to solve small-sized problems; the other two are branch and price-based heuristics for solving large problems. The researchers first provide a mixed integer nonlinear model for routing and pickup and drop-off points selection which they later decompose into a master and subproblem for solving. To validate the performance of their approaches and gather valuable insights about the ridesharing system, the researchers perform numerical experiments on a San Francisco Taxicab dataset. Results show that the approaches are efficient, solving instances with up to 300 nodes within 130 CPU seconds. For these datasets, incorporating flexible meeting points (i.e., pickup and drop-off points) can reduce the total travel time of the rideshare system by 18%. Sensitivity analysis shows that it can also decrease the time passengers wait time for rides by 43%. The methodologies in this study can help transportation planners design more efficient rideshare systems with less waiting, better passenger service, and less travel time.

17. Key Words Rideshare, routing, branch and price, dynamic programming		**18. Distribution Statement** No restrictions.	
19. Security Classif. (of this report) Unclassified	**20. Security Classif. (of this page)** Unclassified	**21. No. of Pages** 48	**22. Price** N/A

NCST

About the National Center for Sustainable Transportation

The National Center for Sustainable Transportation is a consortium of leading universities committed to advancing an environmentally sustainable transportation system through cutting-edge research, direct policy engagement, and education of our future leaders. Consortium members include: University of California, Davis; University of California, Riverside; University of Southern California; California State University, Long Beach; Georgia Institute of Technology; and University of Vermont. More information can be found at: ncst.ucdavis.edu.

Disclaimer

Acknowledgments

This study was funded, partially or entirely, by a grant from the National Center for Sustainable Transportation (NCST), supported by the U.S. Department of Transportation (USDOT) through the University Transportation Centers program. The authors would like to thank the NCST and the USDOT for their support of university-based research in transportation, and especially for the funding provided in support of this project.

NCST

The Ridesharing Routing Problem with Flexible Pickup and Drop-off Points

A National Center for Sustainable Transportation Research Report

November 2022

Maged Dessouky and Zuhayer Mahtab

Daniel J. Epstein Dept. of Industrial and Systems Engineering, University of Southern California

NCST

[page intentionally left blank]

TABLE OF CONTENTS

List of Tables

List of Figures

The Ridesharing Routing Problem with Flexible Pickup and Drop-off Points

EXECUTIVE SUMMARY

Ridesharing can be an effective way of reducing traffic congestion and providing flexible and cheap commutes to passengers. Research has shown that ridesharing systems can increase the efficiency of transportation systems by using unused spaces on vehicles and providing convenient rides to commuters. This is especially applicable to dense urban areas such as Los Angeles and Orange County. One of the major drawbacks of the ridesharing system is the excessive detours drivers have to take to pick up commuters. This not only is an inconvenience to rideshare drivers by also increases travel and waiting times for passengers.

One way this can be reduced is through flexible pickup and drop-off points where passengers walk to a certain meeting point to be picked up or dropped off. Research has shown that this can reduce travel time for all the users in the rideshare system. Although few of the existing literature has considered carpool rideshare system optimization with flexible meeting points, to the best of our knowledge none of the papers have proposed an optimization-based solution procedure and are based on metaheuristics.

In this research project, we propose a dynamic programming-based solution approach and two optimization-based heuristics using Branch and Price to solve the rideshare routing problem with flexible pickup and drop-off points. We consider flexible meeting points and consider the possible use of High Occupancy Lanes (HOV). We also take into consideration the time limits of passengers and drivers. We aim to reduce the travel time for drivers while ensuring the maximum number of commuters will be serviced. We developed a Mixed Integer Nonlinear Program (MINLP) which we decompose into two smaller problems, a route selection problem and a pickup and drop-off point selection problem. To solve the decomposed model, we developed three separate approaches. First is an exhaustive route enumeration algorithm (REA) where we generate all possible feasible routes and their corresponding pickup and drop-off points. The second one is a branch and price-based algorithm with simultaneous routing and pickup and drop-off point selection. The third one is also branch and price algorithm but with sequential routing and pickup and drop-off point selection.

Numerical experiments show that having flexibility in pickup and drop-off points and allowing passengers to walk up to a maximum of 8 minutes can reduce travel time by 18% on average. We also conduct several sensitivity analyses that show that passenger service rate is directly affected by the number of rideshare drivers in the system. These analyses also show increasing the maximum walking tolerance directly results in decreased total travel time, commuter waiting time, and commuter in-vehicle time. Specifically, 8 minutes of passenger walking can decrease the passenger waiting time by 43%. We also found that walking does not affect passenger service rate when the number of drivers in the system is fixed.

Introduction

Traffic congestion is a major problem in large metropolitan areas like Los Angeles and San Francisco Bay area. In 2019, on average a commuter lost about 54 hours or about 1170 USD in monetary terms annually due to congestion nationwide, which amounts to 8.7 billion lost hours in total (Lasley, 2021). Although major infrastructure projects have been taken to alleviate this problem, they cannot keep up with the increase in the usage of personal vehicles (Cheng et al., 2020). On the other hand, these personal vehicles are highly inefficient modes of transport as they usually have a low occupancy rate and take up a lot of space. In 2017, the average vehicle occupancy was only 1.67 (Lasley, 2021). One way of reducing congestion that has shown some promise is ridesharing. Research says that ridesharing can lessen traffic congestion by increasing the efficiency of the transportation system by using empty spaces in vehicles and reducing the need for personal vehicles (Hu et al., 2021; Z. Li et al., 2021).

It is to be noted that we differentiate between commercial ridesharing services which we denote as ride-hailing and carpool ridesharing. In commercial ride-hailing services, drivers provide services which is similar to traditional taxicabs. Here their objective is to earn income. On the other hand, in carpool ridesharing, the drivers are regular commuters, and they take detours to pick up passengers on their way to their destinations. The objective here is travel cost mitigation. Commercial ride-hailing services such as Uber and Lyft have been providing flexible and on-demand rides to commuters for many years and currently are very popular (Hampshire et al., 2017). However, studies show that they exacerbate the congestion problem as they increase deadhead miles and traffic volume (Z. Li et al., 2021). Carpool ridesharing has the potential to solve these problems (Fielbaum et al., 2021).

For a rideshare system to be efficient, one problem that needs to be addressed is how to provide the best routes to the drivers so that they can go to their destination while picking up passengers and delivering them to their destination in the minimum possible time and at the same time satisfying drivers' and passengers' time limits. In this research project, we aim to address this problem.

A drawback of the rideshare system is the excessive detour drivers must take to pick up and drop off passengers. This can be reduced by incorporating flexible pickup and drop-off points, where passengers walk to a common area to be picked up or walk to their destination from the said common area. Research has shown that this can reduce the total travel time of drivers by more than 10% (Fielbaum et al., 2021). We aim to incorporate this walking feature into the rideshare routing problem. For convenience, we denote the pickup and drop-off points as meeting points.

The rideshare routing problem with flexible meeting points is a generalization of pickup and delivery problems (PDP) and dial-a-ride problems (DARP) (B. Li et al., 2016). Numerous papers have been published which discuss the PDP and DARP problems and ways to solve them. Although few of the studies have already considered the rideshare routing problem with flexible meeting points (Fielbaum, 2022; Fielbaum et al., 2021; X. Li et al., 2018), they all propose metaheuristics or insertion heuristics to solve them and consider a limited set of

potential pickup/drop-off points. These proposed methods can have a larger gap to the optimal solution compared to optimization-based heuristics. Also, by limiting the pickup or drop-off points to a discrete set of points, travel time savings can be less compared to a continuous set of pickup/drop-off points. To the best of our knowledge, no optimization-based solution methods have been proposed to solve this problem. Also, no studies have considered a continuous set of pickup/drop-off points. Therefore, in this research project, we aim to provide optimization-based solution approaches to solve the rideshare routing problem with a continuous set of flexible pickup and drop-off points. We aim to develop efficient optimization-based heuristics that can solve medium-sized instances within a reasonable amount of CPU time. We consider the pickup and drop-off time limit for passengers and the destination time limit for drivers. In addition, we also consider the possible usage of High Occupancy Vehicle (HOV) lanes to reduce the travel time. However, the ridesharing system we consider is static and deterministic which may not be robust to changes in request cancellation and traffic congestion.

The rest of the report is organized as follows: In the Literature Review section, we give a brief review of the studies done to date and the state-of-the-art. The Problem Description section describes the problem in mathematical terms. The Solution Methodology section presents the proposed solution approaches. The Numerical Experiments section describes the numerical studies conducted, the dataset on which they are conducted, and the results. Finally, we end the report with a summary and directions for future research in the Conclusions section.

Literature Review

As mentioned previously, the rideshare routing problem is a generalization of the Pickup and Delivery Problem (PDP), which is in turn a generalization of the Vehicle Routing Problem (VRP). There is a vast collection of published research on the VRP. Many solution methodologies have been proposed such as Bender's Decomposition (Alkaabneh et al., 2020) Branch and Price (Christiansen & Lysgaard, 2007; Desrochers et al., 1992), Branch and Cut (Letchford et al., 2007) Branch and Price and Cut (Costa et al., 2019; Desrosiers & Lübbecke, 2011). VRP is a NP-hard problem (Archetti et al., 2011), so various metaheuristics methods have been proposed to approximately solve it such as the genetic algorithm (Baker & Ayechew, 2003), tabu search (Jia et al., 2013), simulated annealing (C. Wang et al., 2015). For a comprehensive review, we redirect the readers to the review of Eksioglu et al. (2009) and Pillac et al. (2013).

The Pickup and Delivery Problem (PDP) expands on the VRP by adding deliveries. Just like the VRP, there is a wide array of literature on this topic. There are many variants of the PDP, the most prominent in the literature is pickup and delivery with time windows (PDPTW). Dumas et al. (1991) developed a branch and price algorithm for the PDPTW. Lu & Dessouky (2004) provided a Branch and Cut algorithm for the problem. Subramanian et al. (2013) developed a Branch and Cut and Price (BCP) algorithm for the problem. They used rounded capacity cuts, bound cuts, and strengthened comb cuts. They were able to solve 4 previously unsolved instances. Cherkesly et al. (2015) also developed a BCP algorithm for the PDPTW but they also added an extra feature namely Last-in-First-out loading (LIFO). Heuristics and metaheuristics have also been used to solve the PDPTW such as the construction heuristic (Lu & Dessouky,

2006; Xiang et al., 2006), simulated annealing (C. Wang et al., 2015), tabu search (Nanry & Wesley Barnes, 2000) and genetic algorithm (Baker & Ayechew, 2003).

The Dial-a-Ride Problem (DARP) is a special variant of the PDPTW where instead of goods, vehicles carry passengers. There is a central depot where it is assumed that an unlimited supply of vehicles is available. The objective is to reduce the total travel time or distance and at the same time reduce the number of vehicles used. There have been quite a few papers published that focus on DARP specifically. Here we discuss a few of the more recent developments. Gschwind & Irnich (2015) provided a Branch and Price methodology for solving the DARP with dynamic time windows. They define a dynamic time window as the upper bound of the difference between the pickup time and drop-off time of a passenger. More recently, Rist & Forbes (2021) provided a new fragment-based formulation that is based on the work of Gshcwind and Irnich. They define a fragment to be a partial route where at the last node a vehicle is empty. They solved the model using a Branch and Cut algorithm and showed that their formulation and method are superior to Branch and Price and Cut based methods. In a later paper, they also developed a Bender's decomposition-based solution approach which outperformed all state-of-the-art solution approaches and was able to solve 11 unsolved benchmark instances of Riedler & Raidl, (2018).

Rideshare Routing Problem

The rideshare routing problem can be perceived as a variation of the DARP, but there is one significant difference. In the rideshare problem, there is no central depot. Instead, the vehicles (or drivers) have their own origins and destinations which can be spread throughout the network. In addition, the drivers have their destination time limits. This adds further complexity to the already difficult dial-a-ride problem. But unlike PDPTW or DARP, rideshare routing problems have not gotten as much attention in the literature. In recent years, the need to solve traffic congestion and the rise and success of commercial rideshare services have shifted the attention of researchers to this problem.

One of the earliest studies on this topic was by Baldacci et al. (2004) who proposed an exact solution approach to solve the to-and-from work carpooling problem. Their method was based on calculating lower bounds of the integer formulation and dynamically generating routes. They were able to solve an instance of up to 250 customers. Some studies focused on the optimal matching of drivers and commuters. For example, Agatz et al. (2011) formulated a bipartite matching integer model to optimally assign passengers to rideshare drivers. In addition, they also included a greedy insertion heuristic. X. Wang et al. (2016) proposed a heuristic solution approach for solving the rideshare routing problem. They used an insertion heuristic, adjust pickup time heuristic, and tabu search to provide good quality routes. They also used 4 sets of valid inequalities to strengthen the formulation. They considered the usage of HOV lanes and how the time savings in HOV lanes affect the optimal routes. Alonso-Mora et al. (2017) proposed an online ridesharing assignment algorithm that can optimize a large-scale ridesharing network in real time. The authors conducted experiments using New York Taxicab dataset considering the full Manhattan road network. They also conducted sensitivity analysis with respect to waiting time, fleet size, vehicle capacity and travel delay. In a later paper,

Simonetto et al. (2019) proposed a new approach for online ridesharing assignment. They showed that by limiting the assignment of a vehicle to at most one customer, they can achieve four times faster execution time without a significant loss in solution quality compared to the approach of Alonso-Mora et al. Their approach can be implemented on less demanding hardware and can be implemented in a multi-company ridesharing scenario. Hasan et al. (2020) developed a Branch-and-Price algorithm for the commute ridesharing problem. Their model had the unique feature of driver-passenger flexibility where if needed any number of passengers can be assigned as drivers. They also used machine learning-based clustering of nodes to make their algorithm more efficient. For a comprehensive review of the rideshare routing problem, we refer the readers to the studies by N. Agatz et al. (2012), Furuhata et al. (2013), and Wang & Yang (2019).

Rideshare Routing with Flexible Pickup and Drop-off Points

Incorporating flexibility in pick-up and drop-off points can result in travel time savings. A few studies have incorporated this feature into their routing problem. Li et al. (2016) devised a dynamic programming-based approach for the multi-meeting point ridesharing problem. Here, instead of directly incorporating walking, they provide a finite set of pickup and drop-off points for each passenger, from which the algorithm chooses the optimal one. Li et al. (2018) developed a tabu search-based solution approach to solve the problem. They did not consider any additional pickup or drop-off points. Instead, they considered other passengers' pickup/drop-off points as the potential pickup/drop-off point for a passenger. They found 2.7%-3.8% time savings as a result of incorporating walking for a 10-passenger rideshare network. They concluded that time savings increases as the rideshare network increases in size. Fielbaum (2022) developed a polynomial time algorithm for selecting stopping points for passengers for a single vehicle case and showed that travel time can be reduced by 20% if detours to pick-up or drop-off passengers can be completely avoided. In a previous paper, Fielbaum et al. (2021) studied the multi-vehicle dynamic rideshare system, where they developed an insertion-based heuristic to solve the passenger-driver assignment problem. The authors considered a finite number of points in the neighborhood of a passenger's originally requested origin or destination point as the potential new pickup or drop-off points. They tested their algorithm on a New York City Taxicab dataset and found that introducing walking to the carpool rideshare can reduce request rejections by 80% and reduce travel time by 10%.

To the best of our knowledge, no studies have considered a continuous set of potential pickup or drop-off points. Also, these studies proposed metaheuristics or insertion heuristics to solve the rideshare routing problem. In this study, we propose three optimization-based heuristics that can solve medium size rideshare networks efficiently. We propose an exact formulation for the problem which we later decompose into a separate routing and pickup/drop-off point selection problem. Our pickup/drop-off point selection problem considers any point that is within the walking distance from the passenger origin/destination as a potential meeting point. We propose a dynamic programming-based heuristic and two branch-and-price-based heuristics. To the best of our knowledge, this is the first study that proposes branch-and-price-based procedures for solving the rideshare routing problem with flexible pickup and drop-off

points. We test the performance of the algorithms on a San Francisco Taxicab dataset. We also conduct sensitivity analysis which will give us valuable insights into the rideshare network.

Problem Description

In this section, we introduce the mathematical model for the rideshare routing and meeting point selection problem and the related notations. In this model, we have a set of m drivers $\{1,2,\dots,m\}$ and a set of n passengers $\{m+1, m+2, \dots, m+n\}$. For this problem, we assume all the passenger requests are known before the planning horizon begins and the travel times between the arcs are known and static. Therefore, we consider a static rideshare system. Each driver and passenger have an origin and a destination. Passenger (Driver) origin nodes are represented by O_P (O_V) and destination nodes are represented by D_P (D_V). We use the same set notation of drivers and passengers to describe the set of origin points for drivers and passengers i.e., $O_V = \{1,2,\dots,m\}, O_P = \{m+1, m+2, \dots, m+n\}$. The destination node sets for drivers are designated by $D_V = \{m+n+1, m+n+2, \dots, 2m+n\}$ and for passengers, it is designated by $D_P = \{2m+n+1, 2m+n+2, \dots, 2m+2n\}$. To represent the problem mathematically, we assume we have a graph $G(N, A)$ where N is the set of nodes i.e., $N = O_V \cup O_P \cup D_P \cup D_V$. $A = \{(i,j): i \in N\ ;\ j \in N\}$ is the set of arcs between node $i, j \in N$. Therefore, our task is to identify sets of optimum routes which are a sequence of arcs. The binary decision variable $X_{i,j,v}$ takes the value 1 if vehicle $v \in V$, goes from node $i \in N$ to node $j \in N$ in the rideshare network and 0 otherwise. Therefore, these decision variables tell us which arcs a vehicle will use in its itinerary.

We denote the time when a node is visited by a driver as t_i $\forall i \in N$. Each passenger has an origin time window represented by $[a_q, b_q]$ where $q \in O_P$ within which they must be picked up. They also have a destination time limit b_q $\forall q \in D_P$ within which they must be dropped off. We assume each driver is traveling to their destination and only takes a detour to pick up a passenger and deliver it to its destination when a passenger is assigned to them. Drivers have destination time windows that are similarly denoted i.e., b_q $\forall q \in D_V$. They also have a ready time a_i $\forall\ i \in O_V$ which is the time they will leave for their destination. We assume passengers are willing to walk a certain distance to go to their pickup point or walk towards their destination if it results in a shorter detour for the driver and minimizes the total travel time. r_i^x and r_i^y are the coordinates of the originally requested pickup or drop-off points and l_i^x and l_i^y are the decision variables for the coordinates of the new pickup or drop-off points where x and y superscript denotes the x and y coordinates and $i \in N$. Using these coordinates, we determine the new travel time $d_{i,j}$ between two nodes $i, j \in N$. Since $d_{i,j}$ is dependent on decision variables (l_i^x, l_i^y) and (l_j^x, l_j^y), it is also a decision variable. Each passenger has a maximum walking distance limit for their origins and destinations which we denote by W_i $\forall i \in O_P \cup D_P$.

The nominal travel speed between two nodes $i, j \in N$ is denoted by $\gamma_{i,j}$. We also assume that for some of the arcs $(i,j) \in A$ $i, j = 1,2,\dots,2m+2n$ an HOV lane can be used if a vehicle has the required number of passengers which will result in a shorter travel time. Thus, a driver will use the HOV lane whenever possible. The minimum number of passengers needed to use the

HOV lane is denoted by H. Each vehicle also has a passenger capacity which is denoted by $C_v\ \forall v \in V$. The HOV lane time savings factor between node $i \in N$ and $j \in N$ is denoted by $\beta_{i,j}$.

Our objective is to find a set of routes for the drivers such that the passengers are picked up and delivered to their desired locations within their respective time windows and the total travel time is minimized while maximizing the passenger service rate. In case there are no compatible passengers the driver will simply go to their respective destinations. We also consider the possibility of having flexible meeting points for passengers which will be introduced later.

We next summarize the decision variables and the parameters and then the model is presented which is denoted as problem P.

Decision Variables

Notation	Description
$x_{i,j,v}$	$\begin{cases}1 \text{ if a vehicle } v \in V \text{ travels from node } i \in N \text{ to a node } j \in N \\ 0 \text{ otherwise}\end{cases}$
t_i	The time when a vehicle visits node $i \in N$
$\alpha_{i,j,v}$	$\begin{cases}1 \text{ if the HOV lane can be used by a vehicle } v \in V \text{ when going from node } i \in N \text{ to node } j \in N \\ 0\ otherwise\end{cases}$
l_i^x, l_i^y	Deviated x and y coordinates of node $i \in O_P \cup D_P$
$d_{i,j}$	Travel time from node $i \in N$ to node $j \in N$ without HOV lanes
$c_{i,v}$	Number of passengers in vehicle $v \in V$ after visiting node $i \in N$

Parameters

Notation	Description
C_v	The capacity of a vehicle $v \in V$ in terms of the number of passengers
a_i	Service start time of node $i \in O_V \cup O_P$
b_i	Service end time of node $i \in O_P \cup D_V \cup D_P$
W_i	Maximum walking distance from node $i \in O_P \cup D_P$
r_i^x, r_i^y	x and y coordinates of node $i \in N$
H	Required number of passengers for HOV lane
$\beta_{i,j}$	The discounted time factor for taking the HOV lane between node i and $j\ \forall i,j \in N$
$\gamma_{i,j}$	The nominal travel speed of vehicles between node i and $j\ \forall i,j \in N$

cd_i Parameter indicating pick up or drop-off of passengers

$$\begin{cases} 1 \text{ if } i \in O_P; \\ -1 \text{ if } i \in D_P; \\ 0 \text{ otherwise} \end{cases}$$

Mathematical Model

$$P = Min \sum_{i \in N} \sum_{j \in N} \sum_{v \in V} (1 - \alpha_{i,j,v}\, \beta_{i,j})\, x_{i,j,v}\, d_{i,j}$$

Subject to

$$\sum_{j \in N} \sum_{v \in V} x_{i,j,v} = 1 \qquad \forall\, i \in N \setminus D_V \qquad (1)$$

$$\sum_{i \in N} \sum_{v \in V} x_{i,j,v} = 1 \qquad \forall\, j \in N \setminus O_V \qquad (2)$$

$$\sum_{j \in N} x_{i,j,v} = \sum_{j \in N} x_{j,i,v} \qquad \forall\, i \in O_P \cup D_P;\ \forall\, v \in V \qquad (3)$$

$$\sum\nolimits_{j \in N} x_{h,j,v} = \sum\nolimits_{j \in N} x_{j,h+n+m,v} \qquad \forall\, h \in O_P\ ;\forall\, v \in V \qquad (4)$$

$$\sum\nolimits_{j \in N} x_{v,j,v} = \sum\nolimits_{j \in N} x_{j,v+n+m,v} \qquad \forall\, v \in O_V \qquad (5)$$

$$\sum_{j \in N \setminus O_V} x_{v,j,v} = 1 \qquad \forall\, v \in O_V \qquad (6)$$

$$\sum\nolimits_{j \in N \setminus D_V} x_{j,v+n+m,v} = 1 \qquad \forall\, v \in O_V \qquad (7)$$

$$x_{i,j,v} = 0 \qquad \forall\, i \in D_v\ ;\forall\, j \in N\ ;\forall\, v \in V \qquad (8)$$

$$x_{i,i,v} = 0 \qquad \forall\, i \in N\ ;\forall\, v \in V \qquad (9)$$

$$t_i + d_{i,j} + M\left(1 - x_{i,j,v}\right) \leq t_j \qquad \forall\, i \in N\ ;\forall\, v \in V \qquad (10)$$

$$t_j \geq a_j \qquad \forall\, j \in O_P \cup O_v\ ;\forall\, v \in V \qquad (11)$$

$$t_j \leq b_j \qquad \forall\, j \in O_P \cup D_p \cup D_V\ ;\forall\, v \in V \qquad (12)$$

$$t_{q+n+m} \geq t_q \qquad \forall\, q \in O_P\ ;\forall\, v \in V \qquad (13)$$

$$c_{i,v} + cd_i + M\left(1 - x_{i,j,v}\right) \leq c_{j,v} \qquad \forall\, i \in N\,; \forall\, j \in N\,; \forall\, v \in V \qquad (14)$$

$$c_{i,v} \leq C \qquad \forall\, i \in N\,; \forall\, v \in V \qquad (15)$$

$$c_{i,v} = 0 \qquad \forall\, i \in O_V \cup D_V\,; \forall\, v \in V \qquad (16)$$

$$\left(l_i^x - l_j^x\right)^2 + \left(l_i^y - l_j^y\right)^2 = \gamma_{i,j}^2\, d_{i,j}^2 \qquad \forall\, i,j \in O_P \cup D_P \qquad (17)$$

$$\left(r_i^x - l_j^x\right)^2 + \left(r_i^y - l_j^y\right)^2 = \gamma_{i,j}^2\, d_{i,j}^2 \qquad \forall\, i \in O_V\,; \forall\, j \in O_P \qquad (18)$$

$$\left(l_i^x - r_j^x\right)^2 + \left(l_i^y - r_j^y\right)^2 = \gamma_{i,j}^2\, d_{i,j}^2 \qquad \forall\, i \in D_P\,; \forall\, j \in D_V \qquad (19)$$

$$\left(r_i^x - r_j^x\right)^2 + \left(r_i^y - r_j^y\right)^2 = \gamma_{i,j}^2\, d_{i,j}^2 \qquad \forall\, i \in O_V\,; \forall\, j \in D_V \qquad (20)$$

$$(l_i^x - r_i^x)^2 + \left(l_i^y - r_i^y\right)^2 \leq W_i^2 \qquad \forall\, i \in O_P \cup D_P \qquad (21)$$

$$\sum_{k \in N} x_{k,i,v}\ cd_k \geq H - M\left(1 - \alpha_{i,j,v}\right) \qquad \forall\, i \in N\,; \forall\, j \in N\,; \forall\, v \in V \qquad (22)$$

$$t_i \geq 0 \qquad \forall\, i \in N\,; \forall\, i \in N\,; \forall\, v \in V \qquad (23)$$

$$x_{i,j,v} \in \{0,1\} \qquad \forall\, i \in N\,; \forall\, j \in N\,; \forall\, v \in V \qquad (24)$$

$$\alpha_{i,j,v} \in \{0,1\} \qquad \forall\, i \in N\,; \forall\, j \in N\,; \forall\, v \in V \qquad (25)$$

$$l_i^x, l_i^y \in \mathbb{R} \qquad \forall\, i \in N \qquad (26)$$

$$c_{i,v} \in \mathbb{Z}^+ \qquad \forall\, i \in N, v \in V \qquad (27)$$

The objective function minimizes the total travel time for all the vehicles on our planning horizon. It considers the deviated pickup and drop-off points for passengers and applies a time discount factor if the vehicle takes the HOV lane. We consider the travel time from node i to node j, $d_{i,j}$ to be a decision variable since the actual pickup and drop-off points for passengers may deviate from their originally requested pickup and drop-off points. Therefore, our problem is a mixed-integer non-linear program (MINLP) with non-linear constraints. Constraints 1-2 are node constraints that ensure that each node is visited exactly once by exactly one vehicle. We note that when we solve the problem, constraints 1-2 may be violated since it is possible that all the passengers may not be served under the given time constraints. Therefore, we relax constraints 1-2 using a relaxation function of the solver and set a penalty for violating constraints 1-2. Hence, our model minimizes the travel time and at the same time minimizes the number of passengers that could not be picked up by the ridesharing system. Constraint 3 is a flow conservation constraint. It states that no vehicle can stay in a node $i \in O_P \cup D_P$ once it arrives at that node. Constraints 4 and 5 assign the origin and destination of a passenger or driver to the same vehicle. Constraint 6 states that all drivers must start their journey from their respective origins. Similarly, Constraint 7 forces all drivers to end their journey at their

respective destinations. Constraint 8 prevents any vehicle to go to any other node after reaching its destination at any point in time. Constraint 9 prevents any circular arcs. Constraints 10 to 13 are time-related constraints that also act as subtour elimination constraints. Specifically, constraint 10 updates the time variable t_i after a vehicle $v \in V$ visits node $i \in N$. Constraints 11 and 12 ensure that passenger and driver origins and destinations are visited within their respective time limits. Constraint 13 is the precedence constraint that ensures that the destinations are visited after the respective origin. Constraint 14 updates the vehicle capacity after it visits a passenger pickup or drop-off node. Constraint 15 is the capacity constraint. Constraint 16 ensures that drivers start their itinerary without any passengers. Constraints 17 to 20 are quadratic constraints that determine the distance between nodes $i, j \in N$ using original or deviated coordinates when applicable. Specifically, constraint 17 determines the travel time between the origin and destination of passengers using their deviated coordinates. Constraint 18 determines the travel time between driver origin and passenger origin using the original coordinates for driver origin point and using the deviated coordinates for passenger origin points. Similarly, constraint 19 determines the travel time between passenger destination and driver destination. Finally, constraint 20 determines the travel time between driver origin-destination pairs. Constraint 21 ensures that deviated pickup or drop-off location is within the passengers' walking limit. Constraint 22 ensures the HOV lane is used whenever a vehicle is carrying the required number of passengers to be eligible for the HOV lane. Finally, constraint 23 is the non-negativity constraint for the time variable t_i. Constraints 24 and 25 are binary constraints for variables $x_{i,j,v}$ and $\alpha_{i,j,v}$. Constraint 27 is integer constraints for the load-variable $c_{i,v}$. One thing to note is that we do not consider the tradeoff between walking time and driving time in the objective function.

Next, we show that the continuous relaxation of the model in the form presented is non-convex. Since the problem is a generalization of PDP, it is NP-hard. Moreover, the nonlinear nature of the model and the non-convexity add another layer of complexity on top of the already NP-hard mixed integer program making solving the model directly using a commercial solver computationally difficult. Therefore, we need to decompose the model to make solving the problem tractable.

Proposition 1: The continuous relaxation of the problem P is non-convex.

Proof: Since the objective function is separable, we can prove that none of the components are convex which will imply the objective function is non-convex. Without loss of generality let us prove that the first component $\left(1 - \alpha_{1,1,1}\, \beta_{1,1}\right) x_{1,1,1}\, d_{1,1}$ is not convex. The Hessian is given by:

$$H = \begin{bmatrix} 0 & 0 & 0 \\ -\beta_{1,1}\, \mathrm{d}_{1,1} & 0 & \left(1 - \alpha_{1,1,1}\, \beta_{1,1}\right) \\ -\beta_{1,1}\, \mathrm{x}_{1,1,1} & \left(1 - \alpha_{1,1,1}\, \beta_{1,1}\right) & 0 \end{bmatrix}$$

Let us determine $\boldsymbol{y}^T\boldsymbol{H}\boldsymbol{y}$ for any $\boldsymbol{y} \in \mathbb{R}^3$

$$[y_1 \quad y_2 \quad y_3]^T \begin{bmatrix} 0 & 0 & 0 \\ -\beta_{1,1}\, d_{1,1} & 0 & (1-\alpha_{1,1,1}\,\beta_{1,1}) \\ -\beta_{1,1}\, x_{1,1,1} & (1-\alpha_{1,1,1}\,\beta_{1,1}) & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$= -y_1 y_3 \beta_{1,1}\, x_{1,1,1} - y_1 y_2 \beta_{1,1}\, d_{1,1} + 2 y_2 y_3 (1-\alpha_{1,1,1}\,\beta_{1,1})$$

This implies $\boldsymbol{y}^T\boldsymbol{H}\boldsymbol{y}$ may not be non-negative. For example, let us take a feasible solution $x_{1,1,1} = 1, d_{1,1} = 1, \alpha_{1,1,1} = 1$ and let $y_1 = 1, y_2 = 1, y_3 = 1, b_{1,1} = .8$. The value of the above expression is then -1.2. Similarly, we can prove that $(1-\alpha_{i,j,v}\,\beta_{i,j})x_{i,j,v}\,d_{i,j}$ is non-convex for any $i \in N, j \in N, v \in V$. Since the objective function is the sum of $(1-\alpha_{i,j,v}\,\beta_{i,j})x_{i,j,v}\,d_{i,j}$ for all $i \in N, j \in N, v \in V$ it is non-convex.■

To form a separate routing problem, we take out constraints 17-21 from P. We define the modified problem P as P'. The objective function of P' is the same as P but the travel time $d_{i,j}$ is now a constant which we determine using the formula $d_{i,j} = \sqrt{\left((r_i^x - r_j^x)^2 + \left(r_i^y - r_j^y\right)^2\right)} \div \gamma_{i,j}$. Therefore, model P' is a rideshare routing problem considering the original pickup and drop-off locations and consists of the constraints 1-16 and 22-25 and 27. The objective function of P' is quadratic, and the constraints are linear. The continuous relaxation of problem P' is now convex. This optimal solution to this decomposition however is not necessarily optimal for problem P due to the routing and pickup/drop-off points selection being separately solved. The meeting points selection model is presented below.

Pickup and Dropoff Point/Meeting Point Selection Problem

From the rideshare routing problem P' we get a set of routes which we then feed into the meeting point selection problem to get the new pickup/drop-off points and the new costs. For this pickup and drop-off point selection model, we denote the routes as $Z = (z_1, z_2, \ldots, z_q)$ where the first node is the driver's origin $z_1 \in O_V$ and the last node is the driver's destination $z_q \in D_V$ and the nodes in between are passenger pickup and drop-offs $z_2, \ldots, z_{q-1} \in O_P \cup D_P$. The meeting point selection model aims to minimize the square of the travel distance of the route by selecting the new pickup and drop-off points while ensuring that those points are within the passengers' walking limits. This model outputs the new pickup and drop-off points and the new total travel time for each route. We denote the models as Q'. The model is given below.

$$(Q')\ Minimize\ \left(l_{z_2}^x - r_{z_1}^x\right)^2 + \left(l_{z_2}^y - r_{z_1}^y\right)^2 + \sum_{\{i=2\}}^{q-2} \left(\left(l_{z_i}^x - l_{z_{i+1}}^x\right)^2 + \left(l_{z_i}^y - l_{z_{i+1}}^y\right)^2\right)$$
$$+ \left(l_{z_{q-1}}^x - r_{z_q}^x\right)^2 + \left(l_{z_{q-1}}^y - r_{z_q}^y\right)^2$$

Subject to

$$\left(l_{z_i}^x - r_{z_i}^x\right)^2 + \left(l_{z_i}^y - r_{z_i}^y\right)^2 \leq W_i^2 \qquad \forall\, i \in O_P \cup D_P$$

In the meeting point selection problem Q', the objective function is the sum of finite quadratic terms. Each of the constraints is also the sum of two quadratic terms. Therefore, the meeting point selection problem Q' is convex. Therefore, this model can be solved rather fast using a commercial solver such as GUROBI. In the next section, we present our solution methodologies.

Solution Methodology

In this section, we present our three different solution methodologies. In the previous section, we decomposed problem P into a routing problem $P'$and a separate meeting point selection problem Q'. However, the routing problem is still NP-hard and quadratic. Therefore, we will decompose the model for problem P'further using Lagrangean relaxation. We will form a set partitioning master problem that will select the best routes and a subproblem that will generate the routes. Our route enumeration approach and the two branch-and-price-based approaches will vary in the structure of the subproblem, and the way routing and meeting point selection problems are integrated. Our first solution approach is a dynamic programming-based route enumeration algorithm that generates the set of all feasible routes, and meeting points for all the passengers in those feasible routes and uses a set partitioning formulation to select the set of best routes. The second and the third one is branch and price algorithms. The difference between the two is in the second one we generate routes and select pickup and drop-off points simultaneously. In the third one, we do this sequentially, i.e., we select the best routes using BP and then select the meeting points. In the next subsection, we present the network preprocessing steps that reduce the complexity of the route generation subproblem by eliminating infeasible arcs. Then we present the set partitioning model and the related notations since this will be used as the master problem in all three approaches. Then, we describe our first solution methodology. Then we describe the Branch and Price algorithm used in the final two approaches. Finally, we present those two approaches and how they integrate routing and meeting point selection.

Network Preprocessing

Before we use any algorithms, we use some network preprocessing that will eliminate infeasible arcs. Since there are $2(m+n)$ nodes in the network graph there can be at most $4(m+n)^2$ arcs. We can reduce the number of possible arcs by doing the following preprocessing steps. Some of the preprocessing steps are adapted for ridesharing from Dumas et al. (1991) and Hasan et al. (2020).

(1) Arcs (i,j) are prohibited if $i \in D_V$ and $j \in N$ since drivers will end their journey once they reach their destination.

(2) Arcs (i,j) are prohibited if $i \in D_P$ and $j = i - m - n$ i.e. arcs from passenger destination to that passenger's origin are prohibited.

(3) Arcs (i, j) are prohibited if $i \in N$ and $j \in O_V$ as all incoming arcs to drivers' origins are infeasible.

(4) Arcs (i, i) are prohibited to prevent any circular arcs for all $i \in N$

(5) Arcs (i, j) are prohibited if $i \in O_V$ and $j \in D_P$ since drivers cannot go to a passenger's destination without picking up that passenger.

(6) Arcs (i, j) are prohibited if $i \in O_P$ and $j \in D_V$ since drivers cannot go to their destinations after picking up and not dropping off a passenger.

(7) Arcs (i, j) are prohibited if $i \in O_V$ and $j \neq i + m + n$ since drivers are not allowed to visit other drivers' destinations.

In addition, we use the following time-based arc elimination techniques. Let $i, j \in O_P$:

(1) Arcs $(i, j + n + m)$ are prohibited if $\max(a_j + d_{j,i}, a_i) + d_{i,j+n+m} + d_{j+n+m,i+n+m} > b_{i+n+m}$

(2) Arcs $(i + n + m, j)$ are prohibited if $\max(a_i + d_{i,i+n+m} + d_{i+n+m,j}, a_j) + d_{j,j+n+m} > b_{j+n+m}$

(3) Arcs (i, j) are prohibited if $\max(a_i + d_{i,j}, a_j) + d_{j,i+n+m} + d_{i+n+m,j+n+m} > b_{j+n+m}$

(4) Arcs (i, j) are prohibited if $\max(a_i + d_{i,j}, a_j) + d_{j,j+n+m} + d_{j+n+m,i+n+m} > b_{i+n+m}$

(5) Arcs $(i + n + m, j + n + m)$ are prohibited if $\max(a_j + d_{j,i}, a_i) + d_{i,i+n+m} + d_{i+n+m,j+n+m} > b_{j+n+m}$

(6) Arcs $(i + n + m, j + n + m)$ are prohibited if $\max(a_i + d_{i,j}, a_j) + d_{j,i+n+m} + d_{i+n+m,j+n+m} > b_{j+n+m}$

Based on driver time limits we prohibit the following arcs. Let $i \in O_V, j \in O_P$:

(1) Arcs (i, j) are prohibited if $\max(a_i + d_{i,j}, a_j) + d_{j,j+n+m} > b_{i+n+m}$

(2) Arcs (i, j) are prohibited if $\max(a_i + d_{i,j}, a_j) + d_{j,j+n+m} > b_{j+n+m}$

(3) Arcs (i, j) are prohibited if $a_i + d_{i,j} > b_{j+n+m}$

Since we use an adjacency graph for the dynamic program, the above arcs are removed from the adjacency lists.

Set Partitioning Formulation for the Routing Problem

In this section, we present the set partitioning formulation for the routing problem. This model will be used as the master problem in all three of our approaches. Let Ω be the set of all feasible routes. we define a new binary variable x_r for each $r \in \Omega$ whose value is 1 if r is used in the solution and 0 otherwise. We also define a new constant $\delta_{i,r}$ whose value is 1 if $i \in N$ is in route r, 0 otherwise. We define the cost of a route as $c_r = \sum_{(i,j) \in A} d_{i,j}$. Thus, the cost of a

route is defined as the sum of the travel times of the arc in that route. We denote the formulation as master problem MP:

$$\text{(MP) } Minimize \ \sum_{r\in\Omega} c_r x_r$$

Subject to

$$\sum_{r\in\Omega} \delta_{i,r}\, x_r = 1 \qquad \forall\, i \in O_P \cup O_V$$

$$x_r \in \{0,1\} \qquad \forall\, r \in \Omega$$

The objective function minimizes the sum of the costs of all the selected routes. The constraint ensures that each driver and passenger origin is visited exactly once. Since not all the passengers may be served, we will relax these constraints while solving the model. The way we generate routes will ensure that all the destinations are in the route if the respective origin is in the route. Therefore, we do not need separate constraints for the destination routes. The second constraint is just the binary constraint for the decision variables.

Route Enumeration Algorithm (Approach 1)

Now we present our first approach which is a dynamic programming-based approach that we call route enumeration algorithm (REA). In this approach, we generate the set of all feasible routes using a dynamic program. Then we generate new meeting points for each of the routes and determine their costs using the meeting point selection problem. Then the routes and their respective costs are fed into the set partitioning formulation which gives us the best set of routes as output. In the dynamic program, for each node, we assign a label that has six elements in it. Each label is denoted by s. Each label of a node $i \in N$ is represented by $s(i) = \{r, R_s, t_s, q_s, Z_s, C_s\}$. The first element r is a partial path from the origin node of the driver up to node $i \in N$. R_s is a set of passengers that have been picked up to node i and not have been dropped off. t_s represents the time of arrival of the vehicle to node i. q_s is the current number of passengers in the vehicle. Z_s is the total travel time without any waiting time to node i from the driver origin. And finally, C_s is the set of passengers that have been picked up and dropped off up to node i. The algorithm is initialized by assigning a label $s(v) = \{\{v\}, \phi, a_v, 0, 0, \phi\}$ for each vehicle $v \in V$ and these labels are put on a stack S. In each iteration, a label is picked from the stack and the algorithm picks the last node i in the partial route r and tries to extend this label to all the nodes in the adjacency list of i. While extending it checks if the expansion is feasible in terms of time limit and capacity. Also, if the number of passengers in the vehicle satisfies the HOV usage requirement it applies a discount factor if available to the travel time. While extending to a passenger destination node it checks if the passenger is in the unserved set R_S. If not, the algorithm does not extend. A time check is performed while extending the label to a passenger origin node that checks if the label can be extended to the respective destination node. Also, while extending to any node the algorithm checks if that node is already in the partial path. That way, we eliminate any possibility of generating a cycle. After extending the label it is again put into the stack. When the algorithm reaches the driver destination node $j \in D_V$, the route is completed. The algorithm then outputs the route and its associated travel

time. In this case, the label is not put into the stack again. This dynamic program terminates when there are no labels in the stack. This algorithm then outputs the set of all feasible paths. Since the number of feasible routes is exponential the route generation dynamic program has an exponential running time. We then feed the routes to the meeting point selection problem Q' to get the new pickup/drop-off points and their new travel times. Then the master problem MP determines the best set of routes using the new travel times. The process of generating feasible routes is given in Figure 1 and the route enumeration algorithm (REA) is summarized in Figure 2.

Now we present the Branch and Price algorithm that will be used in our next two solution approaches.

Branch and Price Algorithm

One drawback of the route enumeration algorithm (REA) approach is the sheer number of feasible routes for even a small-sized network. The Route Enumeration Algorithm (REA) makes the process computationally prohibitive for medium or large-sized networks. One alternative is to consider a small subset of feasible routes at a time and generate routes as needed until we reach the stopping criteria. This is the idea behind the Branch and Price Algorithm (BPA). We use this branch and price algorithm in our approaches 2 and 3. At each iteration, we only consider a small subset of routes which we denote as Ω'. Since we are restricting the master problem (MP) to a small subset of routes we call the new master problem restricted master problem (RMP). The model is given below:

$$\text{(RMP)}\ \ Minimize\ \ \sum_{r\in\,\Omega'} c_r x_r$$

Subject to

$$\sum_{r\in\,\Omega'} \delta_{i,r}\, x_r = 1 \qquad \forall\, i \in O_P \cup O_V$$

$$x_r \in \{0,1\} \qquad \forall\, r \in \Omega'$$

At each iteration, we solve the linear relaxation of the restricted master problem and take the dual variables which we denote as π. We feed these dual variables to the pricing subproblem. The pricing subproblem then provides routes with negative reduced costs which we then add to the restricted set Ω'. If no route with negative route cost can be found, then the linear relaxation of RMP is optimal. We then check the solution of the RMP. If there are any fractional variables, then we do branching on the variables. We branch on the fractional variables until an integer solution is found. The branch and bound scheme will be described later. But first, we describe the pricing subproblem.

Pricing Subproblem

The pricing subproblem aims to generate routes with negative reduced costs. We denote the reduced cost of a route r as $c_r = \sum_{(i,j)\in A} \overline{c_{i,j}}$. where $\overline{c_{i,j}}$ is the reduced cost of the arc (i,j). These are calculated as:

$$\overline{c_{i,j}} = \begin{cases} d_{i,j} - \pi_i \; if \; i \in O_P \cup O_V \\ d_{i,j} \qquad otherwise \end{cases}$$

The pricing subproblem uses a dynamic program-based approach to generate routes similar to REA. Our pricing subproblem is a Shortest Elementary Path Problem with Resource Constraints of Feillet et al. (2004) modified for the ridesharing routing problem. Instead of saving all the labels in a common stack as we did for REA, we keep a list of labels for each node and the stack is filled with nodes to which we will try to extend the labels. The notation and the elements of the label are the same as REA. For each driver, we start with a label $\{\{v\}, \phi, a_v, 0, 0, \phi\}$. At each iteration of the pricing subproblem, the subproblem tries to extend the label to an adjacent route. But here we use dominance criteria to prohibit nonoptimal labels to extend. The dominance criteria are based on Desrochers et al. (1992). Let us have two labels for node i namely $s(i) = \{r, R_s, t_s, q_s, Z_s, C_s\}$. and $s(i)' = \{r', R_s', t_s', q_s', Z_s', C_s'\}$. . Then we say $s(i)$ dominates $s(i)'$ if and only if:

$$R_s \subset R_s', Z_s \leq Z_s' \; and \; t_s \leq t_s'$$

If a label is dominated, we eliminate that label from further extension since it can be proven that a dominated label extends to a nonoptimal path. The proof is given in Desrochers et al. (1992). Whenever the label of a node changes we add that node to the stack E. The algorithm continues until the stack E is empty. After termination the algorithm returns the minimum cost label for every driver destination node $j; j \in D_V$ or $i + n + m; i \in O_V$ from which we extract the minimum cost routes and add them to the set Ω' . The pricing subproblem is given in Figure 3. The label extension procedure is similar to REA and is given in Figure 4.

Algorithm 1 Route generation algorithm

procedure REA(Adjacency list Adj of Graph G)
  For each vehicle $v \in V$ create a new label $\{\{v\}, R_s, t_s, q_s, Z_s, C_s\}$ and put them on stack S
  while S is not empty **do**
    Take a label s from S;
    take the last node $u \in s$;
    while $Adj[u]$ is not empty **do**
      Pick an element w from $Adj[u]$;
      If $q_s \geq H$ then get HOV discount constant $\beta_{u,w}$ else $\beta_{u,w} = 1$;
      if arc (u, w) exists **then**
        if $w \in O_p$ and $t_s + \beta_{u,w} * d_{u,w} \leq b_w$ and $max(t_s + \beta_{u,w} * d_{u,w}, a_w) + d_{w,w+n+m} \leq b_{w+n+m}$ and $max(t_s + \beta_{u,w} * d_{u,w}, a_w) + d_{w,w+n+m} + d_{w+n+m,v+n+m} \leq b_{v+n+m}$ and $q_s < Q$ **then**
          Create a new partial path by appending w to the end;
          Increment $q_{s'}$ by one;
          Add u to $R_{s'}$;
          Set $t_{s'} = max(t_s + \beta_{u,w} * d_{u,w}, a_w)$;
          Set $Z_{s'} = Z_s + \beta_{u,w} * d_{u,w} * \gamma_{u,w}$;
          Create a new label s' and add that to the stack S
        else if $w \in D_p$ **then**
          if $w - n - m$ is in set R_s and $t_s + \beta_{u,w} * d_{u,w} \leq b[w]$ **then**
            Create a new partial path by appending w to the end;
            Reduce $q_{s'}$ by one;
            Remove $w - n - m$ from R_s;
            Update $t_{s'} = t_s + \beta_{u,w} * d_{u,w}$;
            Update $Z_{s'} = Z_s + \gamma_{u,w} * \beta_{u,w} * d_{u,w}$
            Add $w - n - m$ to $C_{s'}$;
            Create a new label s' and add that to S;
          end if
        else if $w \in D_v$ and $t_s + \beta_{u,w} * d_{u,w} \leq b_w$ **then**
          if $R_s = \phi$ **then**
            Create a new path by appending w to the end
            Update $t_{s'} = t_s + \beta_{u,w} * d_{u,w}$
            Update $Z_{s'} = Z_s + \beta_{u,w} * d_{u,w} * \gamma_{u,w}$
            Add the label s' to the output set
          end if
        end if
      end if
    end while
  end while
  return The set of all feasible routes
end procedure

Figure 1. Generating Feasible Routes using Dynamic Programming

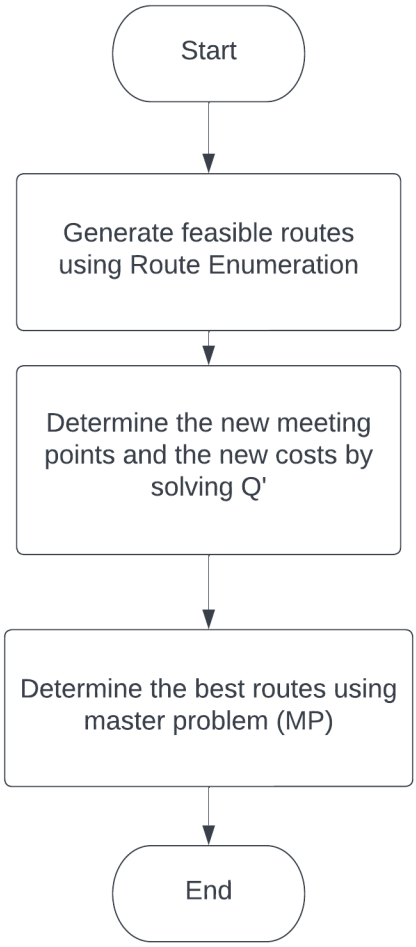

Figure 2. Route Enumeration Algorithm

Algorithm 2 Pricing Sub-problem

```
1: procedure PRICING(driver i,adjacency list)
2:     E ← {i}
3:     Initialize a list S of labels for all the nodes
4:     S[i] = {[i], {φ}, a_i, 0, 0, {φ}}
5:     while E is not empty do
6:         Take an element u from E
7:         for nodes w in the adjacency list of u do
8:             for labels s_u in S[u] do
9:                 s_w ← Extend(s_u, w)
10:                S[w] ← dominate(S[w], s_w)
11:            end for
12:            if labels for node w changed then
13:                Add w to E
14:            end if
15:        end for
16:    end while
17:    return the minimum cost label in S[i + n + m]
18: end procedure
```

Figure 3. Pricing Subproblem

Algorithm 3 Label Extension

procedure PRICING(Label of node u, s_u, node w)
  If $q_s \geq H$ then get HOV discount constant $\beta_{u,w}$ else $\beta_{u,w} = 1$;
  if arc (u, w) exists **then**
    if $w \in O_p$ and v is the corresponding vehicle and $t_s + \beta_{u,w} * d_{u,w} \leq b_w$ and $max(t_s + \beta_{u,w} * d_{u,w}, a_w) + d_{w,w+n+m} \leq b_{w+n+m}$ and $max(t_s + \beta_{u,w} * d_{u,w}, a_w) + d_{w,w+n+m} + d_{w+n+m,v+n+m} \leq b_{v+n+m}$ and $q_s < Q$ **then**
      Create a new partial path by appending w to the end;
      Increment $q_{s'}$ by one;
      Add u to $R_{s'}$;
      Set $t_{s'} = max(t_s + \beta_{u,w} * d_{u,w}, a_w)$;
      Set $Z_{s'} = Z_s + \beta_{u,w} * \overline{c_{u,w}} * \gamma_{u,w}$;
      Create a new label of node w, s_w;
    else if $w \in D_p$ **then**
      if $w - n - m$ is in set R_s and $t_s + \beta_{u,w} * d_{u,w} \leq b_w$ **then**
        Create a new partial path by appending w to the end;
        Reduce $q_{s'}$ by one;
        Remove $w - n - m$ from R_s;
        Update $t_{s'} = t_s + \beta_{u,w} * d_{u,w}$;
        Update $Z_{s'} = Z_s + \gamma_{u,w} * \beta_{u,w} * \overline{c_{u,w}}$
        Add $w - n - m$ to $C_{s'}$;
        Create a new label of node w, s_w;
      end if
    else if $w \in D_v$ and $t_s + \beta_{u,w} * d_{u,w} \leq b_w$ **then**
      if $R_s = \phi$ **then**
        Create a new path by appending w to the end
        Update $t_{s'} = t_s + \beta_{u,w} * d_{u,w}$
        Update $Z_{s'} = Z_s + \beta_{u,w} * \overline{c_{u,w}} * \gamma_{u,w}$
        Create a new label of node w, s_w;
      end if
    end if
  end if
  return Label of node w, s_w;
end procedure

Figure 4. Label Extension Procedure

Branching Scheme

We check the routes from the pricing subproblem to see if there are any negative reduced-cost routes. If found, we add all such routes to Ω' and solve the RMP again. If there are no reduced-cost routes found, then we know our LP relaxation of the RMP is optimal. Then we check if the solution of the RMP is integer. If not, we do branching. We branch based on the flow variable on arcs. We define a flow variable $f_{i,j}$ $\forall (i,j) \in A$. The value of $f_{i,j}$ is the number of times a particular arc (i,j) appears on route subset Ω'. We select all such arcs (i,j) such that $0 < f_{i,j} < 1$ and among these arcs, we select the one with the maximum flow variable. Then we create two branches: one where the value of the flow variable is 0 and another where the value of the flow variable is 1. $f_{i,j} = 0$ is translated to the subproblem by deleting j from the adjacency list of i and penalizing all the routes that use arc (i,j). $f_{i,j} = 1$ is translated to the subproblem by deleting all other nodes except j from the adjacency list of i, deleting j from the adjacency lists of nodes l $\forall l \in N, l \neq i$ and penalizing all the routes that use arc (i,k) $\forall k \neq j$. The newly created branches are put into a branching tree T. At each iteration of the BPA, we take a branch node from the tree and solve the RMP associated with the branch node, take the duals and feed them to the pricing subproblem to generate a new negative reduced costs route. Thus, the branch tree is traversed in a depth-first fashion. The algorithm terminates when the LP relaxation of RMP is optimal, and the solution is integer. We initialize the root node by running the REA for a small amount of time and solving the integer RMP with a short time budget. The idea is that by spending a little more time on the root node we can get close to the optimum solution faster.

Now we describe how routing and meeting point selection is integrated into approaches 2 and 3.

Branch and Price with Simultaneous Routing and Meeting Point Selection (Approach 2)

This is our second proposed approach. In this approach, whenever we generate a new negative reduced cost route in the pricing subproblem, we use the meeting point selection problem Q' to select the new pickup and drop-off points for all the passengers on that route and determine the new cost of the route. Then we add the route and its corresponding cost to the restricted master problem. For notational purposes, we denote this method as 'PreLocBPA'. The procedure is shown in Figure 5.

Algorithm 4 Branch and Price with Simultaneous Routing and Meeting Point Selection

```
Generate the initial routes using REA and select minimum cost routes using RMP
The solution is the current best optimal solution
Initialize the current lower bound to be a large number M
Add the initial optimum routes to R' and create a new RMP with R'
Initialize the branch and bound tree T with the initial RMP
while T is not empty do
    Take a node from T
    Solve the relaxed RMP and get the dual variables π
    Solve the pricing subproblem
    if objective of the RMP<lower bound then
        Update the lower bound
    end if
    if obective of RMP> optimal solution then
        Prune that branch
    end if
    if  routes with negative reduced cost is found then
        Use the meeting point selection problem to generate the new pickup and drop-off points and the new costs
        Add those routes to R' and add their respective costs to the RMP
        Add the new RMP to the tree T
    else if route with negative reduced cost is not found then
        The relaxation of RMP is optimal
        if the solution is integer then
            Update the optimal solution
        else
            Create two branches using the branching scheme and add those to the tree T
        end if
    end if
end while
return the optimal solution
```

Figure 5. Branch and Price with Simultaneous Routing and Meeting Point selection

Branch and Price with Sequential Routing and Meeting Point Selection (Approach 3)

In the third and final approach, we determine the best routes first using their original pickup and drop-off location. And then for each of the selected routes, we use the meeting point selection problem Q' to determine the new pickup and drop-off points and the new costs. The reasoning behind this approach is that since for a medium to large-sized network there are an exponential number of possible feasible routes, even if we consider only a fraction of them at a time, it may take a long time to determine the meeting point for each of the feasible routes. That is why in this approach we determine the routes first and then do the meeting point

selection. For notational purposes, we denote this procedure as ‘PostLocBPA’. The process is summarized in Figure 6.

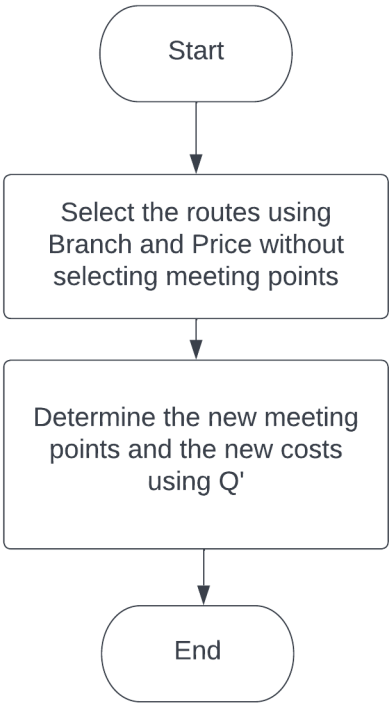

Figure 6. Branch and Price with Sequential Routing and Meeting Point Selection

Numerical Experiments

In this section, we perform numerical tests to determine the effectiveness of our algorithms and to gain some insights into the performance of the rideshare network. First, we do a comparison of the solution quality and computation time among the three proposed methods. Then we conduct a sensitivity analysis where we investigate how the performance changes when we change the number of drivers in the system. We also investigate how the maximum walking time to the meeting point affects the solution and performance parameters. For these analyses, apart from the objective function value, we also use the in-vehicle time for passengers and drivers, their waiting times and the percentage of passengers served as performance measurements. Now we describe the dataset used to do the experiments.

Description of the Dataset

For the experiments, we use the San Francisco taxicab dataset (Piorkowski et al., 2009). It contains the origin and destination of about 650,000 taxicab rides in the San Francisco metro area. We created multiple instances from the dataset. Both driver and passenger origin and destination points are selected from the dataset. Driver origin-destination pairs were selected sequentially from the first half of the dataset. Passenger origin-destination pairs were selected similarly from the second half of the dataset. We set the request arrival horizon to 2 hours. The passenger and driver origin ready times are randomly generated according to a uniform

distribution. The passenger origin due times are generated by adding to the corresponding ready time a fixed time window. The due times for passenger destinations are generated by adding to the corresponding origin due time the product of the direct travel time and a constant μ. Driver destination due times are generated the same way but we multiply by a different constant ν and add the time window. These constants represent how much extension of their direct travel time passengers or drivers are willing to tolerate. The planning horizon is set to be the maximum of the driver's destination due time. We set the penalty for not servicing a passenger to be 1000.

For the High Occupancy Vehicle lanes time discount factor, we generate an $N * N$ matrix, β where each entry is randomly generated using a uniform distribution $\sim Uniform(0,0.1)$. This means vehicles can save up to 10% by using arcs that have an HOV lane between them. These numbers were inspired by Kwon & Varaiya (2008) who found that in California, for every 16-kilometer drive there is a 1.7 minutes time savings on average compared to regular lanes.

For the walking times, we assume that all the passengers are willing to walk a maximum of 10 minutes. Therefore, we set $W_i = 10$ minutes $\forall i \in O_P \cup D_P$. To get the walking distance we multiply the walking time by the nominal walking speed which we set to 5 km/h. We assume all vehicles have the same nominal speed of 40km/h. The table below describes how the parameters are generated and gives the value of the parameters used:

Table 1. Parameters Values Used in Numerical Experiments

Parameter	**Value**
Request Arrival Horizon	120 minutes
Time Window, tw	10 minutes
Driver Origin Ready Time	$Uniform(0, request\ arrival\ horizon \div 2)$
Passenger Origin Ready Time	$Uniform(0, request\ arrival\ horizon)$
Passenger Origin Due Time	Passenger Origin Ready Time+ tw
Driver Destination Due Time	Driver Origin Ready Time+ν*Direct Travel Time+ tw
Passenger Destination Due Time	Passenger Origin Due Time+μ*Direct Travel Time
Passenger Capacity, C	3
Minimum In-vehicle Passenger required to use HOV, H	2
HOV discount factor matrix, β	$Uniform(0,0.1)$
Driver Time Constant, ν	1.5
Passenger Time Constant, μ	1.5
Passenger Max Walking Time, W	10 minutes
The nominal speed of vehicles, γ	40 km\h
Passenger rejection penalty	1000

All the linear programs and integer programs were solved using Gurobi 9.5.0. Since all the passengers cannot be served, we need to relax the constraint in the master problem. For relaxing the constraints, we used the $feasRelax$ function of Gurobi. All the instances were solved on a server with an Intel Xeon Silver processor using 24 cores and 64GB of RAM. All the programs were written in Python 3.6. We note that we tried to solve the MINLP model P directly using Gurobi but none of the instances could be solved even after 6 hours of CPU time. For the Branch and Price based approaches, we run the root node REA for 15 seconds and solve the initial RMP as a mixed integer program with a time budget of 1 hour. Now we compare the performance of our three approaches. First, we show the results for small-sized instances (up to 170 nodes) in Table 2.

Table 2. Performance Analysis of the Three Proposed Approaches for Small Instances

Solution Approach			Route Enumeration Algorithm			PreLocBPA			PostLocBPA		
Instance	**Drivers**	**Passengers**	**CPU Time (sec)**	**Objective (sec)**	**% Served**	**% CPU Time Diff.**	**% Objective Diff.**	**Diff. in % served**	**% CPU Time Diff.**	**% Objective Diff.**	**Diff. in % served**
00d10c20	10	20	10.1	138.2	50%	1479%	0%	0%	1863%	0%	0%
01d20c20	20	20	37.0	267.8	55%	3555%	0%	0%	3856%	0%	0%
02d15c30	15	30	59.6	258.6	57%	3131%	0%	0%	3119%	0%	0%
03d25c30	25	30	172.3	400.6	70%	4230%	0%	0%	4083%	0%	0%
04d20c40	20	40	132.9	389.8	63%	2291%	0%	0%	2149%	0%	0%
05d30c40	30	40	321.2	484.1	70%	2307%	0%	0%	2072%	0%	0%
06d40c40	40	40	1541.2	697.2	85%	4580%	3%	3%	4244%	3%	3%
07d25c50	25	50	447.8	491.2	64%	1892%	-10%	-2%	1844%	-9%	-2%
08d35c50	35	50	2632.7	676.9	84%	7580%	5%	0%	6756%	-1%	-2%
Average						**3449.4%**	**-0.22%**	**0.11%**	**3331.7%**	**-0.7%**	**-0.1%**

In Table 2, we present the solution time, objective value and the percentages of the passengers served for the REA and compare the results with the two Branch and Price approaches. For the other two approaches, we compare the percentage difference in objective value, the percentage difference in CPU time, and the percentage difference served. As we can see from the table, for the first six instances all three methods generate the same solution. For the next three instances, the two branch and price methods generate slightly different solutions. For the instance '06d40c40', the two BPA methods have a 3% gap in objective value, although the percentage of passengers served is also 3% higher. For the next instance '07d25c50', although the objective value is 10% less for PreLocBPA and 9% less for PostLocBPA they both serve 2% fewer passengers. For the last entry in the table, PreLocBPA has a 5% increased objective value and PostLocBPA has a 1% less objective value, but the percentage of passengers served is also 2% less. In terms of CPU time, both branch and price procedures vastly outperform REA, this is to be expected since REA considers all feasible routes whereas BPA systematically generates routes as needed. BPA approaches take 150 to 750 times less computation time than REA. Thus, even though the BPA approaches give a slightly worse solution than REA, they vastly outperform REA in terms of computation time. Therefore, these are better suited for real-life rideshare systems.

For larger instances, since the number of feasible routes is large, REA runs out of system memory. So we only report the absolute values for PreLocBPA and the percentage differences in CPU time, objective value, and percentage served for PostLocBPA. We present the results in Table 3. As we can see in the table, the objective value is slightly higher for sequential meeting point selection in every single instance which is to be expected since, in that approach, we select the pickup and drop-off points after we have selected the best routes. However, in most of the instances, the percentage of passengers served is also higher. In terms of computation time, both approaches perform similarly. There is only a slight difference in computation time. Also, it is to be noted that, the Branch and Price algorithm can solve instances containing up to 300 nodes within a relatively short time. For example, the largest instance with 50 drivers and 100 commuters (300 nodes) was solved within 130 seconds. The highest time taken was 219 seconds which is also a 300-node network but with 70 drivers and 80 passengers. Therefore, BPA approaches are efficient for medium-sized rideshare networks.

Table 3. Performance Analysis of BPA Approaches for Medium to Large Instances

			PreLocBPA			PostLocBPA		
Instance	**Drivers**	**Passengers**	**CPU Time (sec)**	**Objective (sec)**	**% Served**	**% CPU Time Diff.**	**% Objective Diff.**	**Diff. in % served**
10d30c60	30	60	33.2	553.4	63.3%	0%	7%	3.3%
11d40c60	40	60	58.6	768.3	81.7%	-2%	5%	0.0%
12d50c60	50	60	100.5	868.0	80.0%	-4%	8%	1.7%
13d60c60	60	60	144.0	1084.4	81.7%	-1%	0%	0.0%
14d35c70	35	70	50.6	691.6	75.7%	2%	9%	2.9%
15d45c70	45	70	92.6	903.5	80.0%	-4%	0%	1.4%
16d55c70	55	70	135.5	1018.6	80.0%	0%	1%	2.9%
17d65c70	65	70	214.4	1123.6	80.0%	-3%	3%	0.0%
18d40c80	40	80	72.7	772.2	73.8%	-2%	5%	0.0%
19d50c80	50	80	110.3	960.0	75.0%	-1%	3%	2.5%
20d60c80	60	80	157.9	1100.6	75.0%	2%	0%	1.3%
21d70c80	70	80	219.2	1394.2	97.5%	2%	2%	0.0%
22d45c90	45	90	91.5	913.4	76.7%	0%	6%	0.0%
23d55c90	55	90	136.5	1044.6	74.4%	5%	0%	1.1%
24d50c100	50	100	129.7	970.0	74.0%	1%	5%	1.0%
Average						**-0.3%**	**3.6%**	**1.2%**

Effect of Flexible Meeting Points

To show the usefulness of having flexible pickup and drop-off points we compare the results of two approaches: one is our proposed simultaneous routing and meeting point selection branch and price algorithm, and the other is a branch and price algorithm with no walking. The results are shown in Table 4. As we can see, with a maximum walking time of 10 minutes, having flexible meeting points greatly saves travel time with an average savings of 18%. Time savings range from 7% to 33% across all the instances. Although in a few of the instances, the second approach serves more passengers, the time savings is large in comparison. For example, in instance '22d45c50', the no meeting point selection approach serves 1% more passengers but has a 34% increased travel time. From the results, we can see that flexible meeting points can be beneficial in ridesharing systems.

Table 4. Effectiveness of Flexible Meeting Point Selection

			PreLocBPA		No Meeting Point Selection	
Instance	**Drivers**	**Passengers**	**Objective**	**Served**	**% Objective Diff.**	**Diff. in % served**
00d10c20	10	20	138.2	50%	22%	0%
01d20c20	20	20	267.8	55%	12%	0%
02d15c30	15	30	258.6	57%	18%	0%
03d25c30	25	30	400.6	70%	16%	0%
04d20c40	20	40	389.8	63%	20%	0%
05d30c40	30	40	484.1	70%	18%	0%
06d40c40	40	40	715.9	88%	15%	0%
07d25c50	25	50	443.8	62%	26%	2%
08d35c50	35	50	709.2	82%	11%	0%
10d30c60	30	60	553.4	63%	17%	0%
11d40c60	40	60	768.3	82%	23%	0%
12d50c60	50	60	868.0	80%	33%	3%
13d60c60	60	60	1084.4	82%	9%	0%
14d35c70	35	70	691.6	76%	21%	-1%
15d45c70	45	70	903.5	80%	13%	-1%
16d55c70	55	70	1018.6	80%	18%	4%
17d65c70	65	70	1123.6	80%	14%	0%
18d40c80	40	80	772.2	74%	23%	-1%
19d50c80	50	80	960.0	75%	14%	3%

			PreLocBPA		No Meeting Point Selection	
Instance	**Drivers**	**Passengers**	**Objective**	**Served**	**% Objective Diff.**	**Diff. in % served**
20d60c80	60	80	1100.6	75%	8%	3%
21d70c80	70	80	1394.2	98%	17%	0%
22d45c90	45	90	913.4	77%	34%	1%
23d55c90	55	90	1044.6	74%	13%	0%
24d50c100	50	100	970.0	74%	18%	0%
Average Gap					**18%**	**0%**

Sensitivity Analysis

Effect of the Number of Rideshare Drivers

To see the effect of varying the number of rideshare drivers in the system, we took an instance with 45 passengers and varied the number of drivers from 10 to 45. In Figure 7, we can see that as the number of rideshare drivers increases, more commuters are served. With only 10 drivers about 49% or 22 of the 45 passengers were served. But with about 36 drivers 89% or 39 of the 45 passengers were given a ride. This is intuitive, as more rideshare drivers enter the system there are more opportunities for giving rides to commuters while respecting their time limits. The same pattern can be seen in the total travel time in Figure 8. As more drivers enter the system and more commuters are served the total travel time also increases.

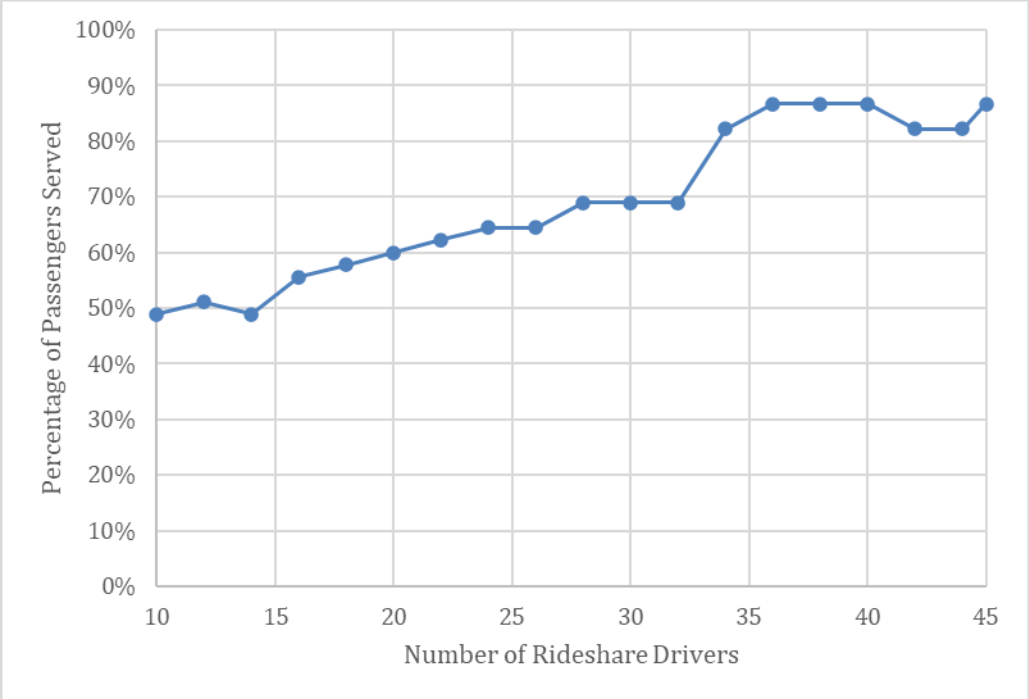

Figure 7. Effect of Varying Rideshare Drivers on the Percentage of Commuters Served

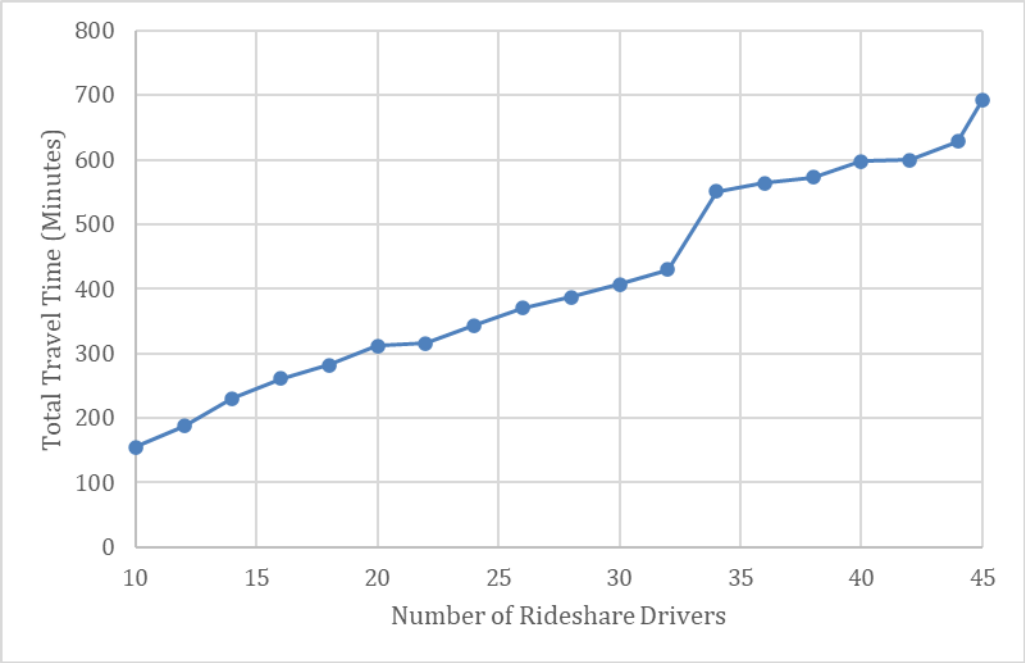

Figure 8. Effect of Varying Rideshare Drivers on the Total Vehicle Travel Time

Effect of Maximum Walking Time

Now we examine how the maximum commuter walking time can change the performance parameters of the system. The results are shown in Figure 9. Here we can see the travel time decreasing as passengers' maximum walking time increases. The maximum time savings is achieved at 8 minutes. However, travel time seems to increase after 8 minutes of maximum walking. This is due to increased driver waiting. Since the meeting point selection problem is solved separately and only minimizes the travel distance, when passengers walk more than 8 minutes, drivers may have to wait at the pickup point for the passenger to arrive which increases the total travel time.

Walking also has a positive effect on the amount of time a passenger spends in the vehicle on average. In Figure 10 we can see the average passenger in-vehicle time (IVT) decreases almost linearly with more walking. With 10 minutes of walking tolerance, average passenger IVT decreases around 30% compared to no walking. A positive effect can also be seen on average passenger waiting time as we can see in Figure 12. In this study, passenger waiting time is defined as the difference between passenger origin ready time and the time the passenger was picked up. Average waiting time decreases almost by 43% when passengers are willing to walk a maximum of 8 minutes compared to no walking. This is intuitive since passengers are willing to walk to a meeting point, they have to wait less for the rideshare drivers to arrive at their location.

However, walking time does not seem to affect the percentage of passengers served much as we can see in Figure 11. From the results of the previous subsection and this, we can conclude that number of rideshare drivers has more of an impact on the percentage of passengers

served than walking time. Since passengers have their service time window, if there are more rideshare drivers, the possibility that one of the rideshare drivers can take a detour to pick up the passenger is higher. But if the number of drivers is low, no matter how much a passenger walks, no rideshare driver may be available that can pick up and drop off that passenger within their time window.

Figure 9. Effect of Maximum Walking Time on Total Vehicle Travel Time (Minutes)

Figure 10. Effect of Maximum Walking Time on Average Passenger in Vehicle Time (Minutes)

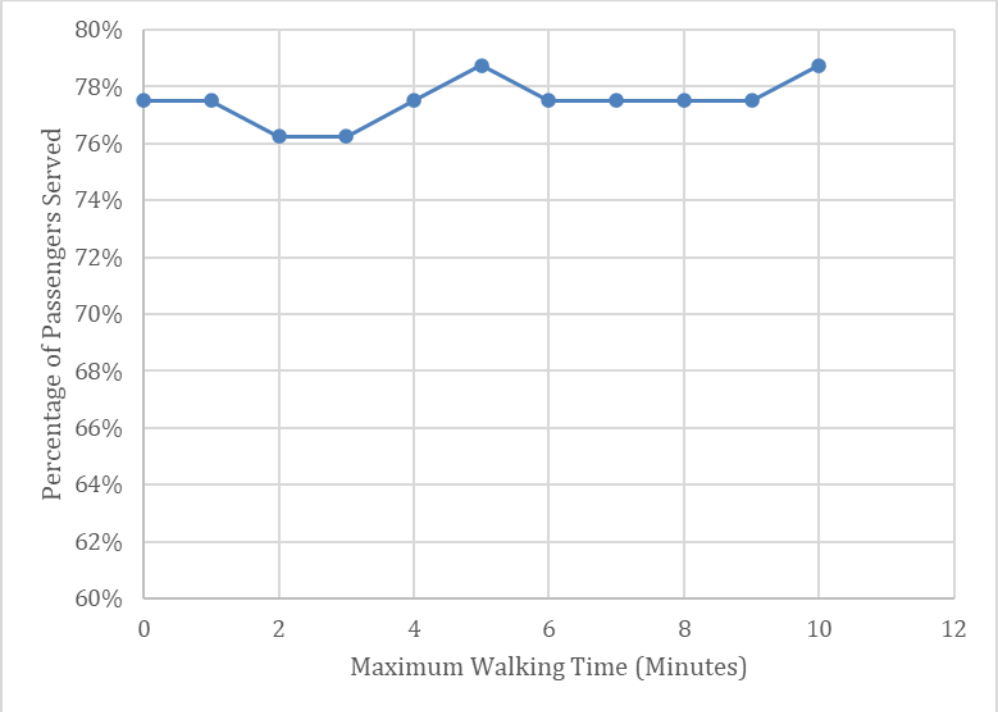

Figure 11. Effect of Maximum Walking Time on Percentage of Passengers Served

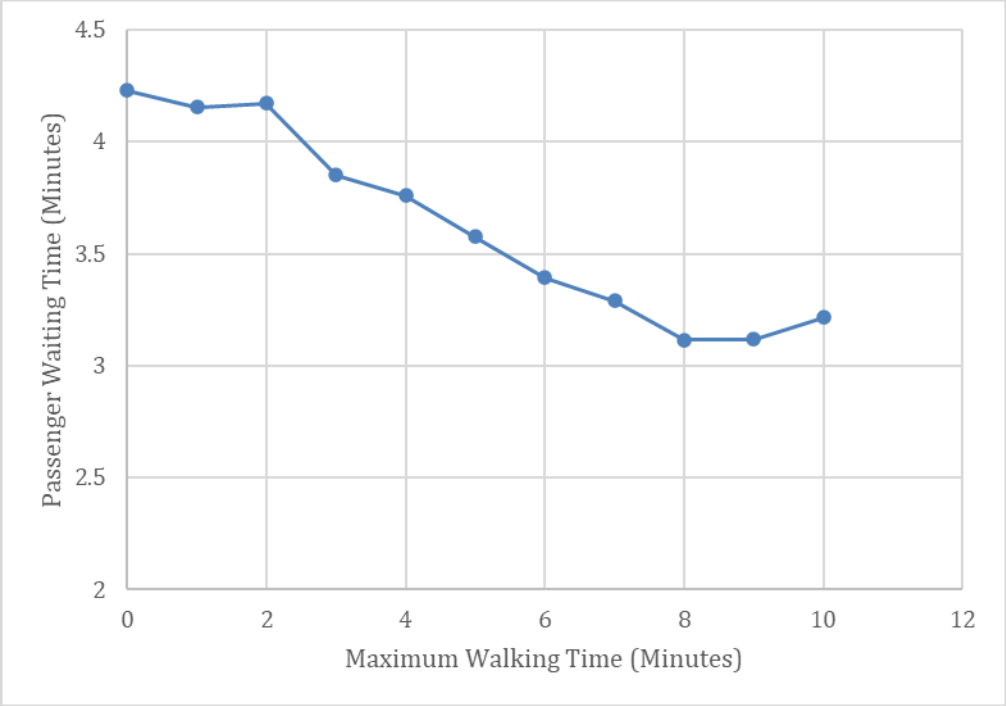

Figure 12. Effect of Maximum Walking Time on Average Passenger Waiting Time (Minutes)

Conclusion

In this study, we developed three approaches for determining the best routes and pickup and drop-off points for rideshare systems. We developed a mixed integer nonlinear model which we decompose into a routing problem and a meeting point selection problem. Later, we decompose the routing problem into a master and subproblem. Our first proposed approach is REA which exhaustively enumerates all feasible routes and determines their corresponding optimum meeting points and uses a set partitioning mixed integer program to determine the set of best routes. The second one is a Branch and Price based approach where we iteratively generate routes and determine optimum meeting points simultaneously until we reach a solution. The third one is a similar approach, but we determine optimum meeting points only for the derived routes.

Numerical results show that, although REA gives us the best solution, due to the exponential number of feasible routes, it is unable to generate routes for medium to large instances. Whereas the other two approaches can solve an instance of up to 300 nodes within 130 seconds. In terms of performance, the two BPA-based approaches perform similarly. The sequential approach is slightly faster, taking 0.33% less computation time on average across all instances. But it also gives a worse solution, with a 3.6% higher total travel time on average.

Results also show that having flexible pickup and drop-off points is beneficial, resulting in 18% total travel time savings on average. Also, an increase in maximum walking tolerance results in lower total travel time, passenger in-vehicle time, and passenger waiting time. The results from this study can help transportation planners design more efficient carpool rideshare systems.

Future research should focus on incorporating meeting point selection directly into the route generation dynamic program which will ensure we get the optimum solution using the Branch and Price algorithm. Also, future research can focus on incorporating valid inequalities which will speed up the computation time of the Branch and Price algorithm even further. Also, uncertainties such as travel and walking times in the rideshare system should be incorporated into future research.

References

Agatz, N. A. H., Erera, A. L., Savelsbergh, M. W. P., & Wang, X. (2011). Dynamic ride-sharing: A simulation study in metro Atlanta. *Transportation Research Part B: Methodological*, *45*(9), 1450–1464. https://doi.org/10.1016/j.trb.2011.05.017

Agatz, N., Erera, A., Savelsbergh, M., & Wang, X. (2012). Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, *223*(2), 295–303. https://doi.org/10.1016/j.ejor.2012.05.028

Alkaabneh, F., Diabat, A., & Gao, H. O. (2020). Benders decomposition for the inventory vehicle routing problem with perishable products and environmental costs. *Computers & Operations Research*, *113*, 104751. https://doi.org/10.1016/j.cor.2019.07.009

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., & Rus, D. (2017). On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, *114*(3), 462–467. https://doi.org/10.1073/pnas.1611675114

Archetti, C., Feillet, D., Gendreau, M., & Speranza, M. G. (2011). Complexity of the VRP and SDVRP. *Transportation Research Part C: Emerging Technologies*, *19*(5), 741–750.

Baker, B. M., & Ayechew, M. A. (2003). A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, *30*(5), 787–800. https://doi.org/10.1016/S0305-0548(02)00051-5

Baldacci, R., Maniezzo, V., & Mingozzi, A. (2004). An Exact Method for the Car Pooling Problem Based on Lagrangean Column Generation. *Operations Research*, *52*(3), 422–439. https://doi.org/10.1287/opre.1030.0106

Cheng, Z. (Aaron), Pang, M.-S., & Pavlou, P. A. (2020). Mitigating Traffic Congestion: The Role of Intelligent Transportation Systems. *Information Systems Research*, *31*(3), 653–674. https://doi.org/10.1287/isre.2019.0894

Cherkesly, M., Desaulniers, G., & Laporte, G. (2015). Branch-Price-and-Cut Algorithms for the Pickup and Delivery Problem with Time Windows and Last-in-First-Out Loading. *Transportation Science*, *49*(4), 752–766. https://doi.org/10.1287/trsc.2014.0535

Christiansen, C. H., & Lysgaard, J. (2007). A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research Letters*, *35*(6), 773–781. https://doi.org/10.1016/j.orl.2006.12.009

Costa, L., Contardo, C., & Desaulniers, G. (2019). Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science*, *53*(4), 946–985.

Desrochers, M., Desrosiers, J., & Solomon, M. (1992). A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *Operations Research*, *40*(2), 342–354. https://doi.org/10.1287/opre.40.2.342

Desrosiers, J., & Lübbecke, M. E. (2011). Branch-Price-and-Cut Algorithms. In J. J. Cochran, L. A. Cox, P. Keskinocak, J. P. Kharoufeh, & J. C. Smith, *Wiley Encyclopedia of Operations Research and Management Science* (p. eorms0118). John Wiley & Sons, Inc. https://doi.org/10.1002/9780470400531.eorms0118

Dumas, Y., Desrosiers, J., & Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, *54*(1), 7–22. https://doi.org/10.1016/0377-2217(91)90319-Q

Eksioglu, B., Vural, A. V., & Reisman, A. (2009). The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, *57*(4), 1472–1483. https://doi.org/10.1016/j.cie.2009.05.009

Feillet, D., Dejax, P., Gendreau, M., & Gueguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, *44*(3), 216–229. https://doi.org/10.1002/net.20033

Fielbaum, A. (2022). Optimizing a vehicle's route in an on-demand ridesharing system in which users might walk. *Journal of Intelligent Transportation Systems*, *26*(4), 432–447. https://doi.org/10.1080/15472450.2021.1901225

Fielbaum, A., Bai, X., & Alonso-Mora, J. (2021). On-demand ridesharing with optimized pick-up and drop-off walking locations. *Transportation Research Part C: Emerging Technologies*, *126*, 103061. https://doi.org/10.1016/j.trc.2021.103061

Furuhata, M., Dessouky, M., Ordóñez, F., Brunet, M.-E., Wang, X., & Koenig, S. (2013). Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, *57*, 28–46. https://doi.org/10.1016/j.trb.2013.08.012

Gschwind, T., & Irnich, S. (2015). Effective Handling of Dynamic Time Windows and Its Application to Solving the Dial-a-Ride Problem. *Transportation Science*, *49*(2), 335–354. https://doi.org/10.1287/trsc.2014.0531

Hampshire, R. C., Simek, C., Fabusuyi, T., & Chen, X. (2017). Measuring the Impact of an Unanticipated Suspension of Ride-Sourcing in Austin, Texas. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.2977969

Hasan, Mohd. H., Van Hentenryck, P., & Legrain, A. (2020). The Commute Trip-Sharing Problem. *Transportation Science*, *54*(6), 1640–1675. https://doi.org/10.1287/trsc.2019.0969

Hu, S., Dessouky, M. M., Uhan, N. A., & Vayanos, P. (2021). Cost-sharing mechanism design for ride-sharing. *Transportation Research Part B: Methodological*, *150*, 410–434.

Jia, H., Li, Y., Dong, B., & Ya, H. (2013). An Improved Tabu Search Approach to Vehicle Routing Problem. *Procedia - Social and Behavioral Sciences*, *96*, 1208–1217. https://doi.org/10.1016/j.sbspro.2013.08.138

Kwon, J., & Varaiya, P. (2008). Effectiveness of California's High Occupancy Vehicle (HOV) system. *Transportation Research Part C: Emerging Technologies*, *16*(1), 98–115. https://doi.org/10.1016/j.trc.2007.06.008

Lasley, P. (2021). *2021 URBAN MOBILITY REPORT*. https://mobility.tamu.edu/umr/report/

Letchford, A. N., Lysgaard, J., & Eglese, R. W. (2007). A branch-and-cut algorithm for the capacitated open vehicle routing problem. *Journal of the Operational Research Society*, *58*(12), 1642–1651. https://doi.org/10.1057/palgrave.jors.2602345

Li, B., Krushinsky, D., Van Woensel, T., & Reijers, H. A. (2016). The Share-a-Ride problem with stochastic travel times and stochastic delivery locations. *Transportation Research Part C: Emerging Technologies*, *67*, 95–108. https://doi.org/10.1016/j.trc.2016.01.014

Li, R.-H., Qin, L., Yu, J. X., & Mao, R. (2016). Optimal Multi-Meeting-Point Route Search. *IEEE Transactions on Knowledge and Data Engineering*, *28*(3), 770–784. https://doi.org/10.1109/TKDE.2015.2492554

Li, X., Hu, S., Fan, W., & Deng, K. (2018). Modeling an enhanced ridesharing system with meet points and time windows. *PLOS ONE*, *13*(5), e0195927. https://doi.org/10.1371/journal.pone.0195927

Li, Z., Liang, C., Hong, Y., & Zhang, Z. (2021). How Do On-demand Ridesharing Services Affect Traffic Congestion? The Moderating Role of Urban Compactness. *Production and Operations Management*, poms.13530. https://doi.org/10.1111/poms.13530

Lu, Q., & Dessouky, M. (2004). An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science*, *38*(4), 503–514.

Lu, Q., & Dessouky, M. M. (2006). A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. *European Journal of Operational Research*, *175*(2), 672–687. https://doi.org/10.1016/j.ejor.2005.05.012

Nanry, W. P., & Wesley Barnes, J. (2000). Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological*, *34*(2), 107–121. https://doi.org/10.1016/S0191-2615(99)00016-8

Pillac, V., Gendreau, M., Guéret, C., & Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research*, *225*(1), 1–11. https://doi.org/10.1016/j.ejor.2012.08.015

Piorkowski, M., Sarafijanovic-Djukic, N., & Grossglauser, M. (2009). *CRAWDAD dataset epfl/mobility (v. 2009-02-24)* [Data set]. Community Resource for Archiving Wireless Data at Dartmouth (CRAWDAD). https://doi.org/10.15783/C7J010

Riedler, M., & Raidl, G. (2018). Solving a selective dial-a-ride problem with logic-based Benders decomposition. *Computers & Operations Research*, *96*, 30–54. https://doi.org/10.1016/j.cor.2018.03.008

Rist, Y., & Forbes, M. A. (2021). A New Formulation for the Dial-a-Ride Problem. *Transportation Science*, *55*(5), 1113–1135. https://doi.org/10.1287/trsc.2021.1044

Simonetto, A., Monteil, J., & Gambella, C. (2019). Real-time city-scale ridesharing via linear assignment problems. *Transportation Research Part C: Emerging Technologies*, *101*, 208–232. https://doi.org/10.1016/j.trc.2019.01.019

Subramanian, A., Uchoa, E., Pessoa, A. A., & Ochi, L. S. (2013). Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery. *Optimization Letters*, *7*(7), 1569–1581. https://doi.org/10.1007/s11590-012-0570-9

Wang, C., Mu, D., Zhao, F., & Sutherland, J. W. (2015). A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup–delivery and time windows. *Computers & Industrial Engineering*, *83*, 111–122. https://doi.org/10.1016/j.cie.2015.02.005

Wang, H., & Yang, H. (2019). Ridesourcing systems: A framework and review. *Transportation Research Part B: Methodological*, *129*, 122–155. https://doi.org/10.1016/j.trb.2019.07.009

Wang, X., Dessouky, M., & Ordonez, F. (2016). A pickup and delivery problem for ridesharing considering congestion. *Transportation Letters*, 1–11. https://doi.org/10.1179/1942787515Y.0000000023

Xiang, Z., Chu, C., & Chen, H. (2006). A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints. *European Journal of Operational Research*, *174*(2), 1117–1139. https://doi.org/10.1016/j.ejor.2004.09.060

Data Summary

Products of Research

One of the main research products of this research will be peer-reviewed journal articles, book chapters and/or conference proceedings targeted toward the transportation science research community, plus supplemental materials such as tables, numerical data used for graphs, etc. The resulting algorithms will be published in peer-reviewed journals.

Data Format and Content

All research products will be available online in digital form. Manuscripts will appear in a common document-viewing format, such as PDF, and supplemental materials such as tables and numerical data will be in a tabular format, such as Microsoft Excel spreadsheet, tab-delimited text, etc.

Data Access and Sharing

All participants in the project will publish the results of their work. Papers will be published in peer-reviewed scientific journals, books published in English, conference proceedings, or as peer-reviewed data reports. Beyond the data posted on USC websites, primary data and other supporting materials created or gathered in the course of work will be shared with other researchers upon reasonable request, at no more than incremental cost and within a reasonable time of the request or, if later, the filing of a patent application covering the results of such research.

All the data used in this research report can be found at Dataverse: https://doi.org/10.7910/DVN/3IWXBJ. This includes the dataset instances, all the data in the tables, and the data for the graphs. The San Francisco Taxicab Dataset can be found at : https://crawdad.org/epfl/mobility/20090224/

Reuse and Redistribution

USC's policy is to encourage, wherever appropriate, research data to be shared with the public through internet access. This public access will be regulated by the university to protect privacy and confidentiality concerns, as well to respect any proprietary or intellectual property rights. Administrators will consult with the university's legal office to address any concerns on a case-by-case basis, if necessary. Terms of use will include requirements of attribution along with disclaimers of liability in connection with any use or distribution of the research data, which may be conditioned under some circumstances.