Stochastic Ridesharing System with Flexible Pickup and Drop-off

February 2024

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
A USDOT University Transportation Center

METRANS Transportation Consortium
USC | CSULB

TECHNICAL REPORT DOCUMENTATION PAGE

1. Report No. NCST-USC-RR-24-04	**2. Government Accession No.** N/A	**3. Recipient's Catalog No.** N/A
4. Title and Subtitle Stochastic Rideshare System with Flexible Pickup and Drop-off Points		**5. Report Date** February 2024 **6. Performing Organization Code** N/A
7. Author(s) Maged Dessouky, Ph.D., 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 (October 2022 – September 2023) **14. Sponsoring Agency Code** USDOT OST-R

15. Supplementary Notes
DOI: https://doi.org/10.7922/G2Q52MZ9
Dataset DOI: https://doi.org/10.6084/m9.figshare.24208827

16. Abstract
Ridesharing can help reduce traffic congestion, greenhouse gas emissions and increase accessibility to transportation in major metropolitan areas across the United States. A robust rideshare system needs to take uncertainties such as traffic congestion and passenger cancellations into account. In this report, the authors propose a data-driven stochastic rideshare system that integrates those sources of uncertainties. Instead of assuming a probability distribution, the approach learns the underlying distribution in travel times and passenger cancellations from historical data. The authors first provide a mathematical model of the problem. Later they propose a stochastic average approximation approach for solving the routing and flexible pickup and drop-off selection problem. They also propose a Branch-and-Price heuristic and Adaptive Large Neighborhood Search-based metaheuristic to solve the underlying rideshare routing problem. To validate the approach, the authors construct test cases based on the New York City taxicab dataset. Numerical results show that the proposed branch and price-based solution approach can efficiently solve small instances while being close to the true optimum. On the other hand, the ALNS-based approach can solve medium to large instances with a small computational time budget while being robust to uncertainties. The proposed approach can help transportation officials and rideshare planners design more robust rideshare systems to alleviate traffic congestion in California.

17. Key Words Rideshare, Routing, Stochastic Optimization		**18. Distribution Statement** No restrictions.	
19. Security Classif. (of this report) Unclassified	**20. Security Classif. (of this page)** Unclassified	**21. No. of Pages** 43	**22. Price** N/A

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

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: the University of California, Davis; California State University, Long Beach; Georgia Institute of Technology; Texas Southern University; the University of California, Riverside; the University of Southern California; and the University of Vermont. More information can be found at: ncst.ucdavis.edu.

Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated in the interest of information exchange. The report is funded, partially or entirely, by a grant from the U.S. Department of Transportation's University Transportation Centers Program. However, the U.S. Government assumes no liability for the contents or use thereof.

The U.S. Department of Transportation requires that all University Transportation Center reports be published publicly. To fulfill this requirement, the National Center for Sustainable Transportation publishes reports on the University of California open access publication repository, eScholarship. The authors may copyright any books, publications, or other copyrightable materials developed in the course of, or under, or as a result of the funding grant; however, the U.S. Department of Transportation reserves a royalty-free, nonexclusive and irrevocable license to reproduce, publish, or otherwise use and to authorize others to use the work for government purposes.

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

Stochastic Rideshare System with Flexible Pickup and Drop-off Points

A National Center for Sustainable Transportation Research Report

February 2024

Maged Dessouky and Zuhayer Mahtab

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

NCST

TABLE OF CONTENTS

List of Tables

List of Figures

Stochastic Rideshare System with Flexible Pickup and Drop-off Points

EXECUTIVE SUMMARY

Traffic congestion and greenhouse gas (GHG) emissions are two of the most pertinent issues in major cities across the United States. Rideshare can be a means to solve both of them. Ridesharing is a service where a driver provides commute to other passengers while driving to their destination. By utilizing unused space in personal vehicles, rideshare can alleviate the traffic congestion problem and reduce GHG emissions. Also, by providing low-cost, flexible, and convenient rides to commuters, rideshare increases the accessibility of economically disadvantaged individuals to transportation.

To increase the adoption of rideshare technology, the service must be efficient. That means drivers must be provided with good quality routes that decrease detours, waiting time, and delay. Passengers must be matched with drivers who will ensure they are picked up and dropped off within their desired time. However, while planning these routes, uncertainties in the transportation system need to be considered. Otherwise, unexpected traffic congestion or passenger cancellations may render these routes suboptimal.

A feature of ridesharing that has shown the potential to decrease detours is flexible meeting points. This is a system where instead of drivers going to the passengers' exact pickup or drop-off location, they meet at a predetermined meeting point. A passenger may have to walk a certain time to the meeting point. Therefore, in this project, we develop a rideshare system with flexible meeting points under travel time and passenger cancellation uncertainties. This system matches passengers to drivers and provides good quality routes to drivers.

To achieve this, we developed a two-stage mixed integer stochastic model. The first stage provides the best routes considering uncertainties. The second stage model calculates the recourse cost under different scenarios. Since the integrated routing and flexible meeting point selection problem is too complex to solve efficiently, we decompose the problem into a separate routing problem and meeting point selection problem.

We then proposed two solution procedures. The first approach uses Branch-and-Price (BP) to solve the stochastic rideshare routing problem. In this procedure, we start with a small subset of feasible routes to select the best routes and iteratively increase the feasible subset size. The second approach uses the Adaptive Large Neighborhood Search (ALNS) procedure, a popular metaheuristic to solve complex optimization problems. This algorithm starts with an initial solution and, at each iteration, modifies it to achieve a better solution until some stopping criteria are reached. We use Sample Average Approximation to generate samples and solve the problem repeatedly until a good-quality solution is reached.

To test the effectiveness of our proposed solution procedures, we construct test instances from the New York Taxicab dataset. We compare the results of our proposed solution procedure

with the Wait-and-See solution. Wait-and-See gives us solutions when we have perfect information about the uncertainties. From the results, we see that Branch-and-Price (BP) provides better solutions than ALNS, although with high computation time. On the other hand, ALNS provides good solutions computationally faster than BP. Our proposed model and solution framework will help rideshare and transportation planners design effective rideshare systems that work well under uncertainty.

Introduction

Rideshare systems can be an effective way of reducing traffic congestion and greenhouse gas (GHG) emissions and increasing accessibility to transport for economically disadvantaged individuals. One way rideshare reduces traffic congestion is by increasing the utilization of private vehicles. According to the 2021 Urban Mobility Report, the average occupancy rate of a vehicle was only 1.67, whereas the capacity of the vehicles is 5-7 (Lasley, 2021). By utilizing the unused capacity, rideshare can reduce traffic congestion. Another way rideshare reduces congestion is by providing convenient and accessible commutes to people, thereby reducing the need for a private vehicle. Although public transport has a higher utilization rate, they are often unavailable in geographically dispersed areas. Moreover, they have limited hours of operation. By reducing the number of vehicles on the road and congestion, rideshare can decrease GHG emissions.

Although commercial ride-hailing services such as Uber and Lyft provide convenient rides to commuters, research has shown they exacerbate traffic congestion and pollution by adding deadhead miles (Z. Li et al., 2021). Moreover, since these services are profit-driven, they require a higher premium to access. In this project, we study ridesharing, which is different from commercial ride-hailing in that the drivers are regular commuters who pick up and drop off other passengers while going to their destination. They are motivated to participate in a rideshare program to reduce their own transportation costs. Therefore, rideshare services provide cheaper rides to passengers. Additionally, rideshare has the potential to be more convenient since the number of potential regular drivers is much higher than commercial ride-hailing drivers. Although ride-hailing and ridesharing may be used interchangeably in the literature, in this project, we make a distinction between them and focus on ridesharing.

For a rideshare program to be effective, passengers need to be matched with drivers with the least inconvenience to both. Drivers should have minimal detour in picking up or dropping off passengers and minimal delay in reaching their own destination. On the other hand, passengers should be picked up and dropped off at their desired time. Therefore, drivers need to be provided with routes that achieve these objectives. Uncertainty is inherent in everyday life, and transportation systems are no different. A sudden change in traffic conditions or passenger cancellation can render these routes ineffective. Therefore, to provide routes that are robust against these uncertainties, we need to consider them when planning rideshare routes. This project addresses how to incorporate uncertainties into the rideshare routing problem.

A significant amount of literature has been published on rideshare routing in deterministic settings. However, there are significantly fewer papers on stochastic rideshare routing. To the best of our knowledge, travel time uncertainty is the only source of uncertainty considered so far in the rideshare routing. However, passenger cancellation is another critical source of uncertainty since it can increase driver detours and in-vehicle time for other passengers. In this project, we consider both sources of uncertainties together. Another way of reducing driver detours is to incorporate flexible pickup and drop-off points. This is a feature where instead of a driver going to a passenger's exact location to pick up or drop off, they meet at a predetermined meeting point. A passenger may have to walk a certain distance to reach the

meeting point, but in doing so, detours can reduced by 17-18% (Dessouky, 2022; Fielbaum, 2022). Therefore, in this project, we study the problem of stochastic rideshare routing with flexible meeting points under travel time and passenger cancellation uncertainty. Moreover, unlike most stochastic vehicle routing literature, where an underlying distribution is assumed, we derive the distribution from historical data. We aim to provide an optimization-based solution method and a metaheuristic-based solution method. Specifically, we address the following research questions:

1. How do we model the uncertainties in the rideshare routing problem?
2. How do we incorporate meeting point selection into the stochastic rideshare routing problem?
3. How do we solve effectively and efficiently the rideshare routing problem with uncertainties?
4. How well do our solution approaches perform compared to the optimal solution for the perfect information scenario?

The rest of the report is organized as follows. In the Literature Review section, we provide a brief overview of problems and solution methodologies studied in the literature on stochastic vehicle routing problems. In the Problem Description section, we describe the problem extensively and provide a mathematical model. In the Solution Procedure section, we provide our two solution approaches. In Numerical Experiments, we show how to construct test instances from historical data and test our solution methodologies on these test instances. We end the report with a Conclusion summarizing the work.

Literature Review

In this section, we highlight the literature on stochastic transportation optimization problems. We first discuss the stochastic vehicle routing literature. Then, we discuss the stochastic pickup and delivery problem and the stochastic dial-a-ride problem, which are generalizations of the vehicle routing problem. We conclude our discussion with stochastic rideshare problems.

Stochastic Vehicle Routing Problems (SVRP)

Vehicle Routing Problems are a generalization of the traveling salesman problems (TSP), where a fleet of vehicles needs to visit a set of customers to meet their demands where a constraint on vehicle load is added to the problem. Since uncertainties are inherent in real-life operations, to make a routing plan robust, it must consider uncertainties that arise in day-to-day operations. In the stochastic VRP (SVRP), one or more of the parameters may be uncertain. The uncertain parameters include travel times, customer demand, waiting times, or a combination (Oyola et al., 2018). However, most of the literature focuses on only a single uncertain parameter. Taş et al. (2014) developed a branch-and-price algorithm for the SVRP with uncertain travel times. Their pricing problem derives the service cost component based on the expected arrival and departure time. They developed a novel ‘Golden Section’ method to estimate the optimum departure time for each vehicle.

Another common uncertain parameter studied in the literature is stochastic or uncertain demands. Christiansen and Lysgaard (2007) developed a branch-and-price method for the SVRP with stochastic demand. Their approach uses a set covering problem for the master problem. Their sub-problem is a graph-based approach that develops routes whose demand variance is less than an upper bound. Their graph-based approach represents every node with expected demand and variance of demand. The arcs of their graph represent the reduced cost of extending the route to that node. By solving a shortest path problem with resource constraints, their subproblem can generate feasible routes in pseudo-polynomial time. Che and Zhang (2023) studied a VRP with simultaneous pickup and delivery with uncertain customer demands. The application of such VRPs lies in closed-loop supply chains and green supply chains. They proposed three recourse actions to nullify the failure to meet customer demands due to uncertainty. They used an Integer L-shaped method to solve the problem. Numerical studies conducted by them showed that their approach resulted in a reduced optimality gap compared to the other literature in SVRP with stochastic demands.

Some literature focused on solving stochastic VRP where multiple parameters are uncertain. For example, Sungur et al. (2010) developed a VRP with probabilistically appearing customers and uncertain travel time. Their approach has two stages: master plan and daily scheduler. The master plan can be considered the first stage of a stochastic program, which develops a vehicle schedule based on the most likely customers and worst-case service time. The daily scheduler is the recourse problem, which updates the master plan daily based on the customers that appear. The model's objective is multi-faceted, minimizing travel times, earliness, and lateness, maximizing the number of customers serviced, and maximizing the similarity between the master schedule and daily schedule. They used an insertion-based algorithm to develop the master plan and a tabu search-based heuristic to improve the master plan. Hou and Zhou (2010) studied a version of VRP where pickup and deliveries are done simultaneously. Their uncertain parameters are stochastic demand and stochastic travel times. They developed a chance-constrained model and developed a genetic algorithm to solve it. They assumed a normal distribution for the uncertain parameters. Saint-Guillain et al. (2017) studied a stochastic VRP with time windows with uncertain customer demands and reveal times. While most of the SVRP literature with uncertain demands assumes a fixed time horizon when the actual demand is revealed, the authors of this paper considered the demand reveal time as uncertain as well. Their recourse problem, the expected cost of the second stage solution, can be computed in polynomial time. To solve the problem, the authors proposed a simulated annealing algorithm. To account for the uncertain reveal time, the authors developed a waiting time strategy where a vehicle waits at a customer until they have to move to the next customer.

Oyola et al. (2018) and Oyola et al. (2017) extensively reviewed stochastic VRPs. Their first paper focuses on model formulation and the different uncertainties considered in the literature. Their second paper, on the other hand, focuses on solution methods. For an extensive review of the subject, we refer the readers to these papers.

Stochastic Pickup and Delivery Problems, and Dial-a-Ride Problems, and Rideshare Routing

Pickup and delivery problems (PDP) are a generalization of the VRP where a fleet of vehicles picks up goods from customers and delivers those goods to other customers. Application of such problem exists in peer-to-peer delivery, public transportation, waste collection, and ambulance scheduling. Just like in VRP, uncertainties exist in travel times, uncertain supply and demand, and uncertain customers. In this subsection, we highlight some of the literature.

Wang et al. (2023) studied a PDP with stochastic and time-dependent travel times. This is a practical aspect of PDP, as travel time usually depends on the time of the day (e.g., peak vs. off-peak hours). The authors estimated the travel times using a step function based on the average speed of vehicles at that time of the day. They estimated the vehicles' expected arrival and departure times at a customer based on methods proposed in other papers. They proposed a branch-and-price-and-cut method to solve the problem. In the pricing problem, the authors developed a novel dominance criterion to eliminate partial routes that are provably sub-optimal. The authors also proposed a heuristic dominance criterion that produces sub-optimal but higher-quality partial routes to speed up the solution time. They used subset row cuts and limited memory subset row cuts.

Dial-a-ride problems (DARP) are a particular variant of PDP where vehicles carry passengers from one place to another instead of goods. Their application includes taxi routing, special mobility services for seniors and disabled people, and flexible transit systems. Their widespread application has led many researchers to study this problem extensively. Lu et al. (2022) studied a stochastic DARP with uncertain travel times on multigraph considering user satisfaction. Their problem was to design a Stochastic DARP problem for a demand-responsive transit system. While most of the literature considers equal penalties for waiting times, ride times, and pickup times violations, the authors considered random variables for the parameters with a known probability distribution. Their multigraph-based approach considers the entire transit network as a graph. The nodes of a graph are the customers, and two arcs connect each pair of nodes; one arc represents the fastest way to get to the other node, while another represents the cheapest way to get to the other node. Their approach determines the sequence of nodes to visit and the suitable arcs to use (cheapest vs. fastest). To solve this problem, authors developed a stochastic average approximation method that uses adaptive large neighborhood search to solve an instance. Zhang et al. (2023) studied both the static and online DARP problems with vehicle-customer coordination. This is a system where the vehicle and passenger meet at a predetermined meeting point, and the meeting is coordinated. They provided an efficient geometric algorithm to solve the coordination problem.

Share-a-ride problems (SARP) can be considered a combination of PDP and DARP, where the same vehicle carries passengers and parcels simultaneously to make the most efficient use of vehicle space. Li et al. (2016) studied the stochastic version of the problem, where they considered both stochastic demands and stochastic travel times individually. The authors formulated both problems as a two-stage stochastic model. Since solving the model directly is computationally intractable, the authors develop an ALNS-based heuristic. They used seven

perturbation operators and ten selection operators. The authors used three approaches to sample the uncertainties systematically: fixed sample size, sample average approximation, and sequential sampling procedure. Using their approach, the authors solved an instance with 75 requests and 200 samples within an hour.

The rideshare routing problem is a generalization of the DARP, where the drivers have their own origin and destination points, starting times, and time limits. Although deterministic rideshare problems have been studied extensively, the literature on stochastic rideshare problems is far less. Long et al. (2018) proposed a stochastic rideshare problem. The authors assumed that each driver carries only one passenger. Therefore, their problem focuses more on the optimal matching of driver and passengers. They considered both uncertain time-independent and time-dependent travel times. The authors analyzed and provided insights on the problem. They showed that there exists an optimal departure time, which is independent of per unit cost of driving and per unit cost of in-vehicle time. Also, they showed that the cost of rideshare increases as the per unit cost and penalty increases.

However, one source of uncertainty that has yet to be considered is random passenger cancellations. If a passenger cancels, the planned route may no longer be optimal, resulting in undesirable detours and delaying the arrival of drivers and passengers. Moreover, to get a robust rideshare plan, it is critical to consider multiple sources of uncertainty together. To the best of our knowledge, the rideshare literature has yet to consider uncertain travel times and passenger cancellations together. In this study, our objective is to develop a rideshare routing model that considers stochastic travel times and passenger cancellations. We discuss the recourse actions that need to be taken. We develop a branch-and-priced-based procedure and an ALNS-based procedure to solve the problem. In addition, we consider a kernel density estimation approach to learn the distribution of uncertainties instead of assuming a distribution since an assumed distribution may not model the uncertainties adequately in a practical situation. To achieve this, we use a Stochastic Average Approximation-based procedure. In addition, we consider flexible meeting points to reduce detours of drivers and passengers further and reduce travel times. We first present a mathematical representation of our model, formulated as a two-stage stochastic mixed integer model. In the next section, we describe our problem mathematically.

Problem Description

This section describes the stochastic rideshare routing problem and presents a mathematical formulation. We have a set of m drivers, which we denote as $D = \{1,2, \dots, m\}$ and a set of n passengers, which we denote with $P = \{m+1, m+2, \dots, m+n\}$. Let O_V be the index set of the origins of the drivers where $O_V = \{1,2, \dots, m\}$ and O_P be the index set of the origins of the passengers where $O_P = \{m+1, m+2, \dots, m+n\}$.

Each driver and passenger also has their respective destination, which we denote using index set R_V as $\{m+n+1, m+n+2, \dots, 2m+n\}$ and index set R_P as $\{2m+n+1, 2m+n+2, \dots, 2m+2n\}$. We have a graph $G = (N, A)$ where N is the node set consisting of all the drivers and passengers' origins and destinations. A is the set of arcs, $A = \{(i,j): i,j \in N\}$. Each

arc $(i,j) \in A$ has an associated travel time $\tau_{i,j}$, which is stochastic with an unknown distribution ϕ. We assume ϕ has a finite support Φ. In the Numerical Experiments section, we will describe how we estimate a distribution for Φ. We also assume that the actual travel times of an arc are revealed when a vehicle travels on that arc.

At the start of the planning horizon, each driver and passenger declares their origin and destination. The origins and destinations are specified by (x,y) coordinates on a 2D Euclidean plane. Therefore, each node $i \in N$ has an associated coordinate, denoted as r_i^x, r_i^y. Each driver $k \in D$ has a travel start time a_k and a maximum ride time limit T_k within which they should reach their destination. Each passenger $p \in P$ has an earliest pickup time a_p and a latest pickup time b_p within which a driver should pick them up. Each passenger also has a maximum ride time limit T_p within which they should be dropped off. Since the travel times are stochastic, we consider soft constraints for these time limits. Violating these time limits will result in a penalty.

We assume a passenger may cancel their request any time before they are picked up. We represent this with parameter χ_i. For $i \in O_P$, $\chi_i = 1$ if the passenger shows up and 0 otherwise. For $j \in R_P$, $\chi_j = -1$ if for the corresponding origin $i \in O_P$, $\chi_i = 1$, and 0 otherwise. We assume this parameter is also stochastic with an unknown distribution ψ_i, which has a finite support Ψ_i. The set of finite support for all $\Psi_i, i \in O_P$ is represented by Ψ. We represent the entire uncertainty set by ξ with a finite support $\Xi = \Phi \times \Psi$. We assume a driver receives information about a passenger cancellation only when the driver goes to that passenger's pickup point. Therefore, if a passenger cancels, then the driver and any passengers inside the vehicle at that time will incur a detour. However, the driver will skip the passenger's drop-off points in that case.

The objective of this problem is to determine the optimum routes for drivers under uncertainties. These routes are the optimum sequence of nodes, from the driver's origin to the matched passengers' origins and destinations and finally to the driver's destination. Therefore, optimum routes are the optimum sequence of arcs. We use binary variables $x_{i,j,k}$ to represent these decisions. $x_{i,j,k}$ is 1 if a driver $k \in D$ takes arc (i,j), 0 otherwise. Variable $t_{i,k}$ represents when a node $i \in N$ is visited by a driver $k \in D$. Since the meeting points are flexible, we represent l_i^x, l_i^y to represent decision variables regarding the coordinates of the new meeting points. $\alpha_{i,j}$ is the distance between node $i,j \in N$. Since we are determining the meeting points, the distance between two nodes is also a decision variable, the values of which depend on the newly determined coordinates. Variable $q_{i,k}$ represents the current load of the driver $k \in D$ after visiting node $i \in N$.

We represent the problem as a two-stage stochastic mixed integer quadratic program. The first stage minimizes the total distance of all the routes and the expectation of the second-stage recourse cost. The output of the first stage model is a set of routes, one for each driver, that minimizes the total distance and recourse cost. The second stage problem is to calculate the recourse cost and recourse action. Table 1 describes the events that may happen due to travel time and passenger cancellation uncertainties and their respective recourse action. We solve a recourse problem for every realization of Ξ. The second stage minimizes the number of

unserved passengers, the waiting time of drivers (if a driver arrives before the earliest pickup time of a passenger), ride time violation of drivers and passengers, pickup time limit violation of passengers, detour due to canceled requests and maximizes the distance savings of not having to visit a canceled passenger's drop-off points for every scenario (or sample) $\xi \in \Xi$.

Table 1. Events Due to Uncertainties and Corresponding Recourse Action

Event	Recourse Action
Driver arrives early	Incur waiting time penalty
Driver picks up a passenger later than latest pickup time of the passenger	Incur pickup time violation penalty
Driver (passenger) reaches his/her destination later than their maximum ride time limit	Incur driver (passenger) maximum ride time violation penalty
Passenger cancels their ride	Incur a detour penalty for going to the passenger's pickup point and skip the passengers drop-off point

Mathematical Model

Sets

P	Set of Passengers
D	Set of Drivers
O_P	Origin of Passengers
R_P	Destination of Passengers
O_V	Origin of Drivers
R_V	Destination of Drivers
N	Set of all nodes

Parameters

C_k	Capacity of the vehicles for $k \in D$
a_p	Earliest pickup time of passenger $p \in P$
a_k	Available start time of driver $k \in D$
b_p	Latest pickup time of passenger $p \in P$
W_i	Maximum walking time limit for passenger origin and destination $i \in O_P \cup R_P$
ζ_i	Nominal walking speed for passenger origin and destination $i \in O_P \cup R_P$
r_i^x, r_i^y	x and y coordinates of node $i \in N$

T_p	Ride time limit for driver or passenger $p \in P \cup D$
M	A large number
χ_i	Parameter indicating pick up or drop-off of passengers

$$\begin{cases} 1\ if\ i \in O_p \\ -1\ of\ i \in R_p \\ 0\ otherwise \end{cases}$$

Variables

$x_{i,j,k}$	1 if driver $k \in D$ travels from node $i \in N$ to node $j \in N$; 0 otherwise
$t_{i,k}$	the time at when a driver $k \in D$ visits node $i \in N$
l_i^x, l_i^y	Deviated x and y coordinates of node $i \in O_P \cup D_P$
$\alpha_{i,j}$	Travel distance from node $i \in N$ to node $j \in N$
$q_{i,k}$	Number of passengers in driver's $k \in D$ vehicle after visiting node $i \in N$

Model SMINP

$$Minimize \sum_{i \in N} \sum_{j \in N} \sum_{k \in D} \alpha_{i,j} x_{i,j,k} + \mathrm{E}_{\xi \in \Xi}[\mathbb{Q}(x, \alpha, \xi)]$$

Subject to

$$\sum_{j \in N} \sum_{k \in D} x_{i,j,k} \leq 1 \qquad \forall\, i \in N \setminus R_V \qquad (1)$$

$$\sum_{i \in N} \sum_{k \in D} x_{i,j,k} \leq 1 \qquad \forall j \in N \setminus O_V \qquad (2)$$

$$\sum_{j \in N} x_{i,j,k} = \sum_{j \in N} x_{j,i,k} \qquad \forall i \in O_P \cup R_P, \quad \forall k \in D \qquad (3)$$

$$\sum_{j \in N} x_{i,j,k} = \sum_{j \in N} x_{j,i+n+m,k} \qquad \forall i \in O_P,\ \forall k \in D \qquad (4)$$

$$\sum_{j \in N} x_{k,j,k} = \sum_{j \in N} x_{j,k+n+m,k} \qquad \forall k \in D \qquad (5)$$

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

$\sum_{j \in N \setminus R_V} x_{j,k+n+m,k} = 1$	$\forall k \in D$	7
$x_{i,j,k} = 0$	$\forall i \in R_V, \forall j \in N, \forall k \in D$	8
$x_{i,i,k} = 0$	$\forall i \in N\ , \forall k \in D$	9
$\left(l_i^x - l_j^x\right)^2 + \left(l_i^y - l_j^y\right)^2 = \alpha_{i,j}^2$	$\forall i, j \in O_P \cup R_P$	10
$\left(l_i^x - r_j^x\right)^2 + \left(l_i^y - r_j^y\right)^2 = \alpha_{j,i}^2$	$\forall i \in O_P, \forall j \in O_V$	11
$\left(l_i^x - r_j^x\right)^2 + \left(l_i^y - r_j^y\right)^2 = \alpha_{i,j}^2$	$\forall i \in R_P,\ \forall j \in R_V$	12
$\left(r_i^x - r_j^x\right)^2 + \left(r_i^y - r_j^y\right)^2 = \alpha_{i,j}^2$	$\forall i \in O_V,\ \forall j \in R_V$	13
$(l_i^x - r_i^x)^2 + \left(l_i^y - r_i^y\right)^2 \leq \zeta_i^2 W_i^2$	$\forall i \in O_P \cup R_P$	14
$l_i^x, l_i^y \in \mathbb{R}$	$\forall i \in N$	15
$x_{i,j,k} \in \{0,1\}$	$\forall i \in N, \quad \forall j \in N, \quad \forall k \in D$	16
$\alpha_{i,j} \geq 0$	$\forall i \in N, \forall j \in N$	17

As mentioned before, the objective function minimizes the sum of the total distance of all the routes and the expected value of the objective of the second stage problem where $\mathbb{Q}(x, \alpha, \xi)$ is the objective function of the second stage problem. Constraints 1 and 2 are flow conservation constraints that state that each node should be visited at most once by one driver. Constraints 3 force each driver to leave any passenger origin and destination. Constraints 4 and 5 assign the same driver to each passenger's and driver's origin and destination. Constraints 6 force each driver to start their journey from their respective origin and constraints 7 force each driver to end their journey at their respective destination. Constraints 8 state that a driver's journey ends when they reach their destination. Constraints 9 prevent circular arcs. Constraints 10 to 13 are meeting point selection constraints. These sets of constraints determine the new distance between nodes based on optimum meeting points. Specifically, constraints 10 determine the distance between all combinations of passenger origins and destinations. Constraints 11 determine the distance between the passengers' pickup meeting points and the drivers' origins. On the other hand, constraints 12 determine the distance between the passengers' drop-off points and the drivers' destinations. Finally, constraints 13 determine the distance between drivers' origins and destinations. Since it is not possible for a driver to go from that driver's origin directly to a passenger's drop-off point without picking up a passenger and go from the passenger's pickup point to the driver's destination without dropping off a passenger, we do not have any constraints to determine the distance between them. Constraints 14 are the walking time limit constraints. They limit the new meeting point to be within the passengers'

walking time limit from the passengers' originally requested pickup or drop-off points. Constraints 15,16 and 17 are domain constraints for the decision variables.

The second stage problem is given below:

$$Minimize\ \mathbb{Q}(x,\alpha,\xi) = \sum_{l=1}^{7} pf_l$$

Subject to

$$pf_1 = \sum_{j\in O_P} \theta_{1,j} \max\{0, \left(\chi_j - \sum_{i\in N}\sum_{k\in D} x_{i,j,k}\right)\}$$

$$pf_2 = \sum_{i\in N}\sum_{j\in O_P}\sum_{k\in D} \theta_{2,j}\ \chi_j\ x_{i,j,k} \max\{0,(a_j - t_{j,k})\}$$

$$pf_3 = \sum_{i\in N}\sum_{j\in R_P}\sum_{k\in D} -\chi_j\theta_{3,j}\ x_{i,j,k} \max\{0,(t_{j,k} - T_j)\}$$

$$pf_4 = \sum_{i\in N}\sum_{j\in R_V}\sum_{k\in D} \theta_{4,j}\ x_{i,j,k} \max\{0,(t_{j,k} - T_j)\}$$

$$pf_5 = \sum_{i\in N}\sum_{j\in O_P}\sum_{k\in D} \theta_{5,j}\chi_j x_{i,j,k} \max\{0,(t_{j,k} - b_j)\}$$

$$pf_6 = \sum_{i\in O_P} \theta_{6,i}(1-\chi_i)\left(\sum_{j\in N}\sum_{k\in D} x_{j,i,k}\alpha_{j,i} + \sum_{j\in N}\sum_{k\in D} x_{i,j,k}\alpha_{i,j}\right)$$

$$pf_7 = \sum_{i\in R_P} (-1-\chi_i)\left(\sum_{j\in N}\sum_{k\in D} x_{j,i,k}\alpha_{j,i} + \sum_{j\in N}\sum_{k\in D} x_{i,j,k}\alpha_{i,j}\right)$$

The objective function of the second stage has seven components $pf_l,\ l = 1,2,\dots,7$. Function pf_1 is the unserved passenger penalty, where $\theta_{1,j}$ is the penalty for not servicing a passenger $j \in O_P$ for passengers who did not cancel. pf_2 is the driver waiting time penalty where $\theta_{2,i}$ is the per unit waiting penalty for driver $k \in D$ and applies only to passengers who did not cancel. pf_3 is the penalty for violating a passenger's maximum ride time limit applied only for serviced passengers who did not cancel. Here $\theta_{3,i}$ is per unit ride time violation penalty for passenger $i \in O_P$. Similarly, pf_4 is the maximum ride time violation penalty for drivers. pf_5 is the passenger pickup time violation penalty. This penalty is applied only to passengers who were serviced and

did not cancel. pf_6 is the detour penalty. Since a driver goes to a passenger pickup, even if the passenger cancels their ride, a detour is incurred. pf_6 calculates the penalty for the detour from a node to the canceled passenger's pickup point to another node. $\theta_{6,i}$ is the detour penalty per distance. Finally, pf_7 is the distance savings due to skipping canceled passengers' drop-off points. Since the routes determined in the first stage contain the canceled passengers' drop-off points, we subtract the distance savings from the objective.

Now, we provide the constraints for the second stage program.

$t_{i,k} + \tau_{i,j} x_{i,j,k} = t_{j,k}$	$\forall i \in N,\ \forall j \in N \setminus O_P,\ \forall k \in D$	18
$\max\left(t_{i,k} + \tau_{i,j} x_{i,j,k}, a_i\right) = t_{j,k}$	$\forall i \in N, \forall j \in O_P, \forall k \in D$	19
$t_{k,k} \geq a_k$	$\forall k \in D$	20
$t_{i+n+m,k} \geq t_{i,k}$	$\forall i \in O_P \cup O_V, \forall k \in D$	21
$q_{i,k} + \chi_j x_{i,j,k} = q_{j,k}$	$\forall i \in N, \forall k \in D$	22
$q_{i,k} \leq C_k$	$\forall i \in N, \forall k \in D$	23
$q_{i,k} = 0$	$\forall i \in O_V \cup R_V, \forall k \in D$	24
$t_{i,k} \geq 0$	$\forall i \in N, \forall k \in D$	25
$q_{i,k} \geq 0$	$\forall i \in N, \forall k \in D$	26

Constraints 18 and 19 update the travel time when a driver $k \in D$ visits a node. Constraints 20 state that drivers start their journey at their respective start times. Constraints 21 state that destination nodes should be visited after the respective origin nodes. Constraints 22 update the load variables after a driver $k \in D$ visits a node. Constraints 23 are capacity constraints for the vehicles. Constraints 24 force each driver to start and end its journey empty. Constraints 25 and 26 are non-negativity constraints for time variable $t_{i,k}$, and load variable $q_{i,k}$. Since all the load parameters are integers, load variables $q_{i,k}$ will also be integers. Hence, the second stage problem is a linear programming problem. In the next section, we describe the solution procedure.

Solution Procedure

In this section, we describe our two proposed solution procedures. We first describe our branch and price-based procedure, which can solve small-to-medium-sized problems efficiently. Next, we describe our adaptive large neighborhood search (ALNS) based procedure, which, although it gives a worse solution than Branch-and-Price, can solve larger instances efficiently. Finally, we end this section with a description of our Sample Average Approximation (SAA) procedure,

which we use to generate samples and determine the stopping criteria. First, we describe how to separate the routing and meeting point selection problem for easier solving.

Separating the Meeting Point Selection (MPS) Problem

The first stage of the rideshare routing with the meeting points selection problem is a quadratically constrained quadratic mixed integer program. To facilitate efficient solving of the problem, we decompose the problem into a separate routing problem and meeting point selection problem. We remove constraints 10 to 15 from the first stage problem to achieve this. Decision variable $\alpha_{i,j}$ then becomes a parameter which we calculate using $\alpha_{i,j} = \sqrt{\left(r_i^x - r_j^x\right)^2 + \left(r_i^y - r_j^y\right)^2}$. Then, the first stage problem becomes a mixed integer linear program, which is easier to solve. Next, we describe the meeting point selection problem.

After solving the first stage problem, we will have a set of routes $Z = (z_0, z_1, \dots, z_{q-1}, z_q)$ where z_0 is the driver's origin and z_q is the driver's destination. $z_1, z_2, \dots, z_{q-1}$ are the passenger's pickup and drop-off points. Our objective is to identify the optimal coordinates of the passengers' meeting points that will be within the passengers' walking time limit and will minimize the total distance of the routes. We denote the meeting points selection problem model as MPS. The problem is given below.

$$(MPS)\ Minimize \left(l_{z_1}^x - r_{z_0}^x\right)^2 + \left(l_{z_1}^y - r_{z_0}^y\right)^2 + \sum_{i=1}^{q-2} \left(\left(l_{z_i}^x - l_{z_{i+1}}^x\right)^2 + \left(l_{z_i}^y - l_{z_{i+1}}^y\right)^2\right) + \left(l_{z_{q-1}}^x - r_{z_q}^x\right)^2 + \left(l_{z_{q-1}}^y - r_{z_q}^y\right)^2$$

Subject to

$$\left(l_{z_i}^x - r_{z_i}^x\right)^2 + \left(l_{z_i}^y - r_{z_i}^y\right)^2 \leq \zeta_i^2 W_i^2 \qquad \forall\, i = 1,2,\dots,q-1$$

The above formulation is quadratically constrained quadratic program which is convex. Therefore, it can be solved efficiently using a commercial solver such as Gurobi. We will integrate the MPS into various stages of our proposed Branch-and-Price and ALNS algorithms. Next, we describe the branch-and-price procedure.

Branch and Price

Branch and Price is an iterative solution procedure where, in each iteration, we solve a relatively small problem with only a subset of feasible solutions (Barnhart et al., 1998). The concept behind Branch and Price is since the feasible solution space of the problem is too large to solve efficiently, we can start with a very small subset of the solution space. By cleverly adding more feasible solutions at each iteration, we can achieve the optimal solution with much less computation time.

We first decompose the problem into a master problem and a pricing subproblem. The objective of the master problem is to determine the best set of routes. The objective of the pricing problem is to generate feasible routes. These feasible routes are added to the master problem and solved again. However, we only consider a small subset of feasible routes at each iteration. The master problem is thus denoted as the restricted master problem (RMP). At every iteration of BP, we solve the linear relaxation of the RMP and get the duals. We then feed the duals to the pricing problem, generating feasible routes with negative reduced costs. If any route with a negative reduced cost is found, we add it to the RMP and solve the RMP again. If not found, then the current linear relaxation is optimal. In that case, we check the solution to see if it is integer. If the solution is integer, we update the incumbent solution. If not, we create two branches by fixing the value of a variable to 0 and 1. The branching scheme used is similar to our previous work for the deterministic version of the problem (Dessouky, 2022). If a linear relaxation of the RMP has a higher objective value than the incumbent solution, we prune that branch. We continue this iterative procedure until the branching tree is empty. The incumbent solution is returned as the best solution. Figure 1 describes the procedure. We next describe our master problem. Then, we will describe the pricing problem.

Algorithm 1 Branch and Price Approach

```
Generate the initial routes and select minimum cost routes using MP
The solution is the incumbent solution
Initialize the current lower bound to be a large number M
Add the initial best routes to Ω′ and create a new RMP with Ω′
Initialize the branch and bound tree H with the initial RMP
while H is not empty do
    Take a RMP from H
    Solve the relaxed RMP and get the dual variables π, ρ and σ
    if Objective of the relaxed RMP<lower bound then
        Update the lower bound
    end if
    if Objective of relaxed RMP> incumbent solution then
        Prune that branch
    end if
    Solve the pricing subproblem
    if  Routes with negative reduced cost is found then
        Use MPS to generate the new pickup and drop-off points and update the new costs
        Calculate the recourse cost of the routes
        Add those routes to Ω′ and add their respective costs to the RMP
        Add the new RMP to the tree H
    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 incumbent solution
        else
            Create two branches using the branching scheme and add those to the tree H
        end if
    end if
end while
return The incumbent solution
```

Figure 1. Branch and Price Algorithm

Master Problem

The objective of the master problem is to select a set of best routes (one for each driver) from the set of feasible routes. We denote the master problem as MP. We have a set of feasible routes, which we denote as Ω. We have a set of binary variables $y_r \ \forall r \in \Omega$. y_r is 1 if route r is selected and 0 otherwise. We denote the cost of the route as c_r. It is the summation of the total distance of the route and the expected sum of $pf_2, pf_3, \dots, pf_7$ of the second stage problem described above. Therefore, it includes all the first-stage and second-stage costs of the route except for the unmet passenger penalty cost since we can only calculate once we have selected a subset of feasible routes. $\delta \in \mathbb{B}^{|\Omega| \times N}$ is a matrix that indicates which nodes are in

each route. Therefore, $\delta_{r,i} = 1$ means $i \in N$ is in route r and 0 otherwise. The formulation MP is given below:

$$(MP)\ minimize \sum_{r\in\Omega} y_r c_r + E_{\xi\in\Xi}\left[\sum_{r\in\Omega}\sum_{i\in O_P} \theta_{1,i} \max\left(0, \chi_i - \delta_{r,i} y_r\right)\right]$$

Subject to

$$\sum_{r\in\Omega} \delta_{r,i} y_r = 1 \qquad \forall i \in O_V \qquad 27$$

$$\sum_{r\in\Omega} \delta_{r,i} y_r \leq 1 \qquad \forall i \in O_P \qquad 28$$

$$y_r \in \{0,1\} \qquad \forall r \in \Omega \qquad 29$$

The objective function minimizes the sum of the costs of the route and the expected cost of the unserved passenger penalty. Constraints 27 state that each driver should be included in the solution exactly once. Constraints 28 state that each passenger should be included in the route at most once since all passengers cannot be served. Finally, constraints 29 is the domain constraint for variables y_r.

In each iteration of the BP, we solve the MP using only a subset of feasible routes. We denote the subset as Ω'. Then we denote the master problem as restricted master problem (RMP).

Pricing Problem

As mentioned before, the objective of the pricing problem is to generate feasible routes with negative reduced costs. We denote negative reduced cost as $\overline{c_r} = \sum_{(i,j)\in r} \overline{c_{i,j}}$ where $\overline{c_{i,j}}$ is the reduced cost of arc (i,j). The following equations determine the reduced cost of an arc.

$$\overline{c_{i,j}} = \begin{cases} c_{i,j} - \pi_i \ \forall i \in O_V \\ c_{i,j} + \rho_i - \theta_{1,i} E_{\xi\in\Xi}[\sigma_i] \ \forall i \in O_P \end{cases}$$

Here, $c_{i,j}$ is the cost of arc (i,j). π_i are the duals associated with constraints 27. ρ_i are the duals associated with 28. σ_i are the duals associated with the linearization of the maximization function in the second part of the objective function of MP. We get one dual per sample, and we take the expectation.

Our pricing problem is a labeling algorithm. We start by initializing one label per vehicle. We represent a label ending at node $i \in N$ as $L_i = \{r(L), c_r(L), t(L), q(L), C(L), I(L), V(L)\}$. $r(L)$ is the partial route associated with label L that ends in node i. $c_r(L)$ is the reduced cost associated with the label. $t(L)$ is the associated time of visiting the last node i. $q(L)$ is the current number of passengers on the vehicle. $C(L)$ is a set containing passengers whose pickup point has been visited but whose drop-off points have not been. $I(L)$ is another set that

contains all the nodes that have been visited so far. Finally, $V(L)$ contains the nodes explored so far in the labeling process.

We start the labeling process by initializing labels $L_i = \{[i], 0, a_i, 0, \phi, \phi, \{i\}]\}\ \forall i \in O_V$ for each driver origin and add node i to a last-in-first-out (LIFO) stack. In every iteration of the pricing problem, we take a node i from the stack, and for every label L_i of i, we try to extend the label to an adjacent node of i in the graph G. To accelerate the process, we prevent suboptimal labels from extending using a dominance criteria.

Let us say we have two labels L_i and L_i' associated with node i. Then we say label L_i dominates L_i' if $C(L) = C(L'), I(L) = I(L')$ and $c_r(L) \leq c_r(L')$. This is different from the dominance criteria proposed in the deterministic literature since, due to stochasticity, the triangle inequality of time and distance cannot be established. After extending a label to node i we check the dominance criteria to all other labels of node i. If the new node is dominated, we do not add that label, and it is eliminated from further extension. On the other hand, if the new label dominates other labels, the other dominant labels are eliminated from further extension.

The label extension procedure is described in Figure 2. We continue until the associated driver's destination has been reached. The partial routes are completed at that point, and the label extension terminates. The pricing problem then checks for any routes with negative reduced costs. If found, they are returned to the RMP. The procedure is shown in Figure 3.

Algorithm 3 Label Extension

procedure PRICING(Label of node u, L_u, node w, driver k associated with label L_u)
  if arc (u, w) exists **then**
    if $w \in O_p$ **then**
      if $q \leq Q_k$ **then**
        Create a new partial route $r(L_w)$ by appending w to the end;
        Increment $q(L_w)$ by 1;
        Add w to $C(L_w)$;
        Add w to $I(L_w)$;
        Set $t(L_w) = max(t(L_u) + \mathbb{E}[\tau_{u,w}], a_w)$;
        Set $c_r(L_w) = c_r(L_u) + \alpha_{u,w} + \theta_{2,w} max(0, a_w - (t(L_u) + \mathbb{E}[\tau_{u,w}])) + \theta_{5,w} max(0, t(L_u) + \mathbb{E}[\tau_{u,w}] - b_w)$;
        Create a new label of node w, L_w;
      end if
    else if $w \in R_p$ **then**
      if $w - n - m$ is in set $C(L_u)$ **then**
        Create a new partial route $r(L_w)$ by appending w to the end;
        Decrement $q(L_w)$ by 1;
        Remove $w - n - m$ from $C(L_w)$;
        Add w to $I(L_w)$;
        Set $t(L_w) = t(L_u) + \mathbb{E}[\tau_{u,w}]$;
        Set $c_r(L_w) = c_r(L_u) + \alpha_{u,w} + \theta_{3,w} max(0, t(L_u) + \mathbb{E}[\tau_{u,w}] - T_w)$;
        Create a new label of node w, L_w;
      end if
    else if $w \in R_v$ **then**
      if $C(L_u) = \phi$ **then**
        Create a new route $r(L_w)$ by appending w to the end
        Add w to $I(L_w)$;
        Set $t(L_w) = t(L_u) + \mathbb{E}[\tau_{u,w}]$;
        Set $c_r(L_w) = c_r(L_u) + \alpha_{u,w} + \theta_{4,w} max(0, t(L_u) + \mathbb{E}[\tau_{u,w}] - T_w)$;
        Create a new label of node w, L_w;
      end if
    end if
  end if
  return A label of node w, L_w;
end procedure

Figure 2. Label Extension Procedure

Algorithm 2 Pricing Sub-problem

procedure PRICING(driver i, adjacency list of Graph G)
  Initialize a list of labels $Labels$ for all the nodes
  $L_i = \{[i], 0, a_i, 0, \phi, \phi, \{i\}\}$
  $Labels[i] \leftarrow L_i$
  $Stack \leftarrow i$
  while $Stack$ is not empty **do**
    Take a node u from $Stack$
    for nodes w in the adjacency list of u **do**
      for labels L_u in $Labels[u]$ **do**
        if w is not in the explored set $V(L_u)$ **then**
          Add w to the explored set $V(L_u)$
          $L_w \leftarrow Extend(L_u, w)$
          if if any label $L'_w \in Labels[w]$ does not dominate L_w **then**
            add L_w to $Labels[w]$
          else if if any label $L'_w \in Labels[w]$ is dominated by L_w **then**
            remove L'_w from $Labels[w]$
            add L_w to $Labels[w]$
          end if
        end if
      end for
      if labels for node w changed **then**
        Add w to $Stack$
      end if
    end for
  end while
  return the negative reduced cost labels in $Labels[i + n + m]$
end procedure

Figure 3. Pricing Sub-Problem

Adaptive Large Neighborhood Search (ALNS)

ALNS is an extension of Large Neighborhood Search (LNS) introduced by Shaw (1998). It was proposed to solve pickup and delivery problems with time windows (PDPTW) by Ropke and Pisinger (2006). Since then, ALNS has been used in other areas such as VRP, VRPTW, Share-a-ride problem, and rideshare routing with promising results. ALNS differs from LNS by using multiple selection and removal heuristics, which are simple but fast to execute, instead of a single heuristic. Moreover, ALNS also uses simulated annealing as an acceptance criterion for a solution. By probabilistically accepting worse solutions, ALNS can escape local optima. We propose this method as an alternative to our branch-and-price-based method. Although BP gives a good solution, its computational time makes solving larger instances computationally prohibitive. Using simple local search heuristics, ALNS can give a good solution within a reasonable computation time.

We represent a solution S using a set of routes that has a dimension equal to the number of drivers present in the problem. In other words, $S = \{Z_i, i \in O_V\}$ where Z_i is the route of driver

i. We keep a pool B of unserved passengers currently not in the solution S. We start the algorithm by deriving an initial solution using an insertion heuristic. In each iteration, we choose a selection and a perturbation operator from a list of operators using the roulette wheel mechanism. The weights of the roulette wheel are updated adaptively based on the operator's performance and the number of times a particular operator has been used. Selection operators select a random number of passengers β from S and put them into pool B. The Perturbation operator first removes passengers in B from S, then takes passengers from B and tries to insert them into S using an insertion operator. Our algorithm then checks the quality of the solution. If it is better than the current global best solution, then we update the global best solution. We update the current solution if it is worse than the global best but better than the current solution.

On the other hand, if it is worse than the global best and current solution, the solution is still accepted if it meets the simulated annealing acceptance criteria. Finally, we check if the stopping criteria have been met. If the stopping criteria are met, the global best solution is returned. The algorithm continues otherwise.

Next, we describe the selection and perturbation operators, the acceptance and stopping criteria, the choice of selection and perturbation operators, and the initialization procedure.

Selection Operators

In our proposed ALNS procedure, we use five selection operators. They are described below:

1. **Random Selection:** In random selection, we randomly select β number of passengers from S to put them into B.
2. **Worse Distance Selection:** The operator selects β passengers from S whose pickup and drop-off points that have the worst distance from the neighboring nodes in the route. To achieve this, we first rank the passengers in S according to their distance to the origin and destination node from the neighboring nodes and select the bottom β to be put into B.
3. **Highest Cancellation Removal:** This operator selects β passengers for removal who have the highest cancellation rates in the samples. Since passenger cancellation results in detour and delay for the driver and all other passengers assigned to that driver, removing a passenger who is likely to cancel can result in a better solution.
4. **Worse Recourse:** This operator selects β passengers whose inclusion in the solution S results in the highest increases in the recourse cost. To calculate the marginal increase in recourse cost, we first remove a passenger from S and calculate the recourse cost. We then compare the recourse cost of S with and without the passenger. After getting all the marginal recourse costs, we select β passengers with the highest marginal recourse costs.
5. **Shaw Removal:** This selection operator was proposed by Shaw (1998). It uses a relatedness measure between nodes to remove nodes that are closely related since, by removing closely related nodes, it may be possible to diversify the search. We use the following equation to determine the relatedness measure between two passengers.

$$relatedness(i,j) = \kappa\left(\alpha_{i,j} + \alpha_{i+n+m,j+n+m}\right) + (1-\kappa)\left(t_{i,k} - t_{j,k} + t_{i+n+m,k} - t_{j+n+m,k}\right)$$

Where κ is a weight parameter. We calculate the relatedness measure for every pair of passengers $i, j \in O_P$ and keep them in a relatedness matrix. Then we randomly select a passenger and put its closely related passenger into B until we have selected β passengers.

Perturbation Operators

Perturbation operators remove the requests from B that are in S using insertion heuristics. It removes the passengers' origin and destination from the solution, then tries to insert it back into the solution using some criteria that vary among perturbation operators. Note that set B contains some passengers who are in the solution and some who are not in the solution (i.e., unserved). Therefore, perturbation tries to insert unserved passengers as well. While inserting, we check the current capacity of the vehicle. Only feasible positions (without exceeding vehicle capacity) are considered. We use four perturbation operators in our proposed ALNS heuristics. These perturbation operators were inspired by Li et al. (2016) and Ropke and Pisinger (2006) adapted to the rideshare routing problem. We describe the operators below:

1. One-by-One: This operator removes each passenger's origin and destination one by one and then inserts them into the best possible position. First, it determines the best position to insert a passenger's origin, and then it tries to insert the passenger's destination into the best feasible position (not before the origin).
2. Balanced one-by-one: Like one-by-one, this operator removes passengers' OD pairs one by one, then ranks all the routes by their respective costs and inserts them only into the best position of the lowest cost route. Therefore, it significantly reduces the search time for the best possible position.
3. Second Best: It is similar to one-by-one but inserts a node into the second best position to diversify the search.
4. Tabu Insertion: In tabu insertion, when nodes are inserted into their best position, the node's position and the vehicle it has been assigned to are added to a tabu list. If, in the next iteration, tabu insertion is used as a perturbation operator, it will prevent the node from being inserted into the same position as the previous iteration to diversify the search.

Acceptance and Stopping Criteria

As previously mentioned, if, at any iteration, a solution is better than the global best or previous solution, it is accepted. If it is worse than the global best or previous solution, it is accepted probabilistically using a simulated annealing criteria. In this case, we generate a random number between 0 and 1; if that random number is lower than the following exponent, it is accepted:

$$e^{-\frac{current\ solution\ value - previous\ solution\ value}{Temp}}$$

Here, $Temp$ is the temperature which is updated using $Temp = .9999Temp'$, where $Temp'$ is the temperature of the previous iteration.

We use a maximum iteration count and a convergence score called g-score for the stopping criteria. Initially, the g-score is set to 0. At every iteration, the g-score is updated using the following equation:

$$g - score = 0.99 * g - score' + \frac{|f_{best} - f_{current}|}{\min(f_{best}, f_{current})}$$

Where f_{best} is the objective value of the global best solution, and $f_{current}$ is the objective value of the current solution. Therefore, the g-score represents the convergence rate. We continue the ALNS algorithm until the maximum iteration number is reached or the g-score exceeds 0.5.

Choice of Selection and Perturbation Operators

We choose the selection and perturbation operator using a roulette wheel mechanism. In every iteration, the selection and perturbation operator is selected with probability:

$$\frac{weight_i}{\sum weight_j}$$

The weights are updated on every iteration using the following equation:

$$weight_i = \omega\, weight_i + (1 - \omega)\frac{score_i}{times_i}$$

Where $weight_i$ is the operator's weight, $score_i$ is the score of the operator i, and $times_i$ is how many times an operator has been used. ω is a parameter that determines the update rate. We update the scores based on the operator's performance at every iteration. We also increment the usage counter of that operator. If, at any iteration, a new global best solution is found, the score of the selection and perturbation operator is increased by η_1. If a better solution than the previous iteration is found, the score is increased by η_2. On the other hand, if a worse solution is found but is accepted using simulated annealing, the score is increased by η_3. We set the score increments so that $\eta_1 > \eta_2 > \eta_3$. Initially, the scores are set so that every operator is equally likely to be chosen.

Initialization

The performance of ALNS is dependent upon the quality of the initial solution. To get a good initial solution, we use an insertion heuristic. At every iteration, we randomly choose a passenger to be inserted into the solution. We first check the best position for the pickup node at every vehicle and every position. After finding the best position for the pickup node, we find the best position for the drop-off node. We find the best position for the passenger by finding the solution cost before and after a passenger is included. The passenger is included if it results in a lower solution cost. We repeat this for every passenger. If some passengers have not been included in the solution at the end of the insertion heuristics, we add them to the unserved set B.

Figure 4 summarizes the entire solution procedure. Next, we describe our Sample Average Approximation (SAA) procedure.

Algorithm 4 Adaptive Large Neighborhood Search

Generate the initial routes using an insertion heuristic
The solution is the incumbent solution S_{best} and the objective value is f_{best}
Iteration counter $iter = 0$ and $g - score = 0$
while $iter < maxiter$ or $g - score < 0.5$ **do**
    Choose a selection and perturbation operator using a roulette wheel mechanism
    Use the selection and perturbation operator to generate a new solution S'
    Use MPS to generate the new meeting points and the objective value f'_s
    if $f'_s < f_{best}$ **then**
        Update the incumbent solution and the best objective value
        Increment the score of the selection and perturbation operator by η_1
    else if $f'_s < f_s$ **then**
        Update the current solution S and f_s
        Increment the score of the selection and perturbation operator by η_2
    else if Solution is accepted via simulated annealing mechanism **then**
        Update the current solution S and f_s
        Increment the score of the selection and perturbation operator by η_3
    end if
    Update the usage counter of selection and perturbation operator by 1
    if $iter$ is a multiple of 100 **then**
        Update the g-score
    end if
    Increment $iter$ by 1
end while
return The best solution S_{best} and the objective value f_{best}

Figure 4. Adaptive Large Neighborhood Search Algorithm

Sample Average Approximation (SAA)

SAA is an iterative sampling procedure where we estimate the expected value of the stochastic program using a sampling problem. In this procedure, we repeatedly solve the rideshare routing with flexible meeting points problem in Λ batches. In each $\lambda = 1,2, \dots, \Lambda$ batch, we generate Γ samples that are i.i.d. We solve the problem using those Γ samples via ALNS or BP algorithm and get the solution S_λ^Γ and the objective value f_λ^Γ. We then evaluate the statistical lower bound $f^\Gamma = \sum_{\lambda'=1}^{\lambda} f_{\lambda'}^\Gamma$ and the variance of the lower bound $var^2(f^\Gamma) = \frac{1}{\lambda(\lambda-1)} \sum_{\lambda'=1}^{\lambda} (f_{\lambda'}^\Gamma - f^\Gamma)^2$. The solution is S_λ^Γ evaluated using $\Gamma'(\Gamma' \gg \Gamma)$ i.i.d. samples. From this, we evaluate the statistical upper bound $f^{\Gamma'} = \frac{1}{\Gamma'} \sum_{\gamma=1}^{\Gamma'} f_\gamma$ where f_γ is the objective value of the solution evaluated on sample γ. We also estimate the variance of the upper bound $var^2(f^{\Gamma'}) = \frac{1}{\Gamma'(\Gamma'-1)} \sum_{\gamma=1}^{\Gamma'} (f_\gamma - f^{\Gamma'})^2$. We denote the best upper bound found as f_{best}. We then evaluate the SAA gap and the variance of the gap estimator. SAA gap is the gap between the statistical lower bound and the true optimal solution of the stochastic problem, which we evaluate using

$f_{best} - f^{\Gamma}$. The variance of the gap estimator is the summation of the variance of the lower bound and the variance of the upper bound. At each λ, we check if the SAA gap and the variance are sufficiently small. If so, we generate Γ'' i.i.d. samples and evaluate the λ solutions. The best solution which gives the lowest objective value on those Γ'' samples is returned as the best solution of the entire problem. If the gaps are not sufficiently small even after Λ batches, we increase the size of Γ and Γ' and start over. Figure 5 describes our algorithm.

Algorithm 5 Sample Average Approximation

while SAA gap and variance of the gap estimator is not sufficiently small **do**
  for $\lambda = 1, 2, ..., \Lambda$ **do**
    Generate Γ random i.i.d. samples
    Solve the problem using BP or ALNS using those Γ samples to get the solution and objective value
    Update the lower bound which is the average of solution value of λ batches
    Update the variance of the lower bound
    Generate Γ' random i.i.d samples
    Evaluate the solution on these Γ' solutions
    Calculate the upper bound and the variance of the upper bound
    Calculate the SAA gap and the variance of the gap estimator
  end for
  if SAA gap and the variance of the gap estimator are not sufficiently small **then**
    Increase Γ and Γ' and solve for Λ more batches
  end if
end while
Generate Γ'' random i.i.d samples
Evaluate the Λ optimal solution on these Γ'' samples
Return the solution with lowest objective value

Figure 5. Sample Average Approximation Procedure

Numerical Experiments

In this section, we describe the numerical experiments conducted to verify the effectiveness and efficiency of our proposed solution procedures. We first describe the dataset and instance generation method. Then, we present the results of the two methods and how they perform compared to a solution with perfect future information.

Description of the Dataset and Sample Generation

In this research project, we use the New York Taxicab and Limousine Commission dataset (New York (N.Y.). Taxi And Limousine Commission, 2019). This dataset contains information about over a billion rides throughout New York City from 2009 to the current year. It contains yellow, green, and black taxicab data and data about commercial rideshare applications such as Uber and Lyft. It contains the coordinates of the origin and destination points and ride start and end times. We use this extensive dataset to construct our test instances.

For instance generation, we only focus on commercial rideshare data from 2018 to 2023. This dataset differs from yellow or green taxicab data as it contains the zone number instead of the origin and destination coordinates. New York City authority divided the entire city into 261 zones. The names of the zones are stored in a separate file.

To construct instances, we first generate the distances. We first decode the names of the zones into coordinates using Python's GeoPy library. After getting the coordinates of the zones, we determine the distances using the geodesic function and store the distances into a 261×261 matrix. We also store the coordinates in a separate vector.

We use Kernel Density Estimate (KDE) to generate the samples of the travel times. We first gather historical travel times data. We then use KDE to train the model on historical data. KDE can then generate any number of new samples using a function call. Unfortunately, the dataset does not contain any data about passenger cancellations history. So, we generate the data. We first use a vector of size 1000 to store data about 1000 distinct passengers. For each passenger, we randomly generated data for 1000 days. Each passenger has a 0.1 probability of canceling their rides each day. We then again used KDE to train the models for each passenger to generate new samples.

We randomly select some passengers from the passenger vector to generate test instances. We randomly assign the pickup and drop-off zones. Similarly, for drivers, we randomly assign them origin and destination zones. We denote an instance by an instance number, followed by the number of drivers and passengers. For example, '00d5c5' indicates it is the 0^{th} instance with five drivers and five passengers.

We must assign each driver their journey start time and maximum ride time extension. We set the journey start time randomly within the planning horizon. To set the driver's maximum ride time extension, we take 1.5 times the expected travel time between the driver's origin and destination. We must also assign each passenger their desired pickup time window and maximum ride time extensions. We set the pickup time window randomly within the planning horizon. The passenger's ride time limit extension is set by multiplying the expected travel time between their origin and destination by 1.5. We describe the parameters below in Table 2.

Table 2. Parameters for Instance Generation

Parameter	Value
Request Arrival Horizon	710 minutes
Time Window, tw	10 minutes
Driver Journey Start Time	$Uniform(0, request\ arrival\ horizon)$
Passenger Pickup Start Time	$Uniform(0, request\ arrival\ horizon)$
Passenger Pickup Due Time	Passenger Origin Ready Time+ tw
Driver Maximum Ride Time Limit	1.5*Expected Direct Travel Time
Passenger Maximum Ride Time Limit	1.5*Expected Direct Travel Time
Vehicle Capacity, C	3
Passenger Max Walking Time, W	10 minutes
The Nominal Speed of Vehicles, γ	40 km\h

Results of the Numerical Experiments

Before we present the results of the numerical experiments, we present the values of the parameters used. For simplicity, we assume penalties such as ride time extension penalty, driver waiting time, and passenger pickup delay penalty are equal for all passengers and drivers. In Table 3, we describe the values of the penalties used.

Table 3. Second-stage Penalties and Their Values

Penalty	Value
Unserved Passenger Penalty $\theta_{1,i}$	350
Driver Waiting Time Penalty $\theta_{2,i}$	0.05
Passenger Ride Time Extension Penalty $\theta_{3,i}$	0.1
Driver Ride Time Extension Penalty $\theta_{4,i}$	0.1
Passenger Pickup Time Violation Penalty $\theta_{5,i}$	0.1
Detour penalty $\theta_{6,i}$	0.2

For the Sample average approximation method, the values of the parameters used are given in Table 4.

Table 4. Parameter Values of Sample Average Approximation

Parameter	Value
Acceptable SAA Gap	5%
Acceptable Variance of the Gap Estimator	10%
Γ	100
Γ'	1000
Γ''	1000
Batch Size, Λ	10
Γ Increment Value	100
Γ' Increment Value	1000

The proposed methods were implemented using Python. The linear programs and MPS were solved using the Gurobi solver. All tests were conducted on an Intel Xeon computer with 32 GB of RAM. We next present the results of the Branch and Price.

Results of Branch and Price

We present the results in Table 5. Our proposed BP approach can solve instances with up to 70 nodes. Any instances with higher nodes become computationally prohibitively expensive to solve. As the number of nodes increases, the execution time generally increases, although not monotonically. For example, instance '03d10c15' took 1627 seconds to solve, whereas a larger instance '04d15c15' took 863 seconds. This is due to the convergence of SAA. Since the samples generated are random, this may cause the SAA to converge in different numbers of batches. Looking at the number of passengers included in the route, we can see that in all instances, all the passengers were included (even if some of them might cancel in some of the scenarios). This is due to two factors:

1. In instance generation, we set the probability of canceling to 10%, which is relatively low. This means that most passengers do not cancel across all samples.
2. The penalty for not serving a passenger is higher than the detour penalty. Therefore, the algorithm will try to include as many passengers as possible to obtain a lower objective value.

Table 5. Results of the Branch and Price Algorithm

Instance	No. Drivers	No. Passengers	Execution Time (seconds)	Objective Value	Average Number of Passengers Served
00d5c5	5	5	210.33	83.52	5
01d5c10	5	10	267.63	149.82	10
02d10c10	10	10	486.03	85.18	10
03d10c15	10	15	1627.55	232.92	15
04d15c15	15	15	863.33	203.69	15
05d15c20	15	20	12674.9	191.59	20

Next, we present the results of the ALNS algorithm.

Results of Adaptive Large Neighborhood Search (ALNS)

We report the parameters used in Table 6. The results of the numerical experiments are shown in Table 7. Compared to the BP approach, ALNS can solve much larger instances, solving instances with up to 200 nodes. However, the execution time does not monotonically increase with larger instances. Since ALNS is a metaheuristic and we use SAA as the sampling approach,

the convergence of the procedure is unpredictable. However, except for one instance '06d20c20', all other instances have been solved within an hour of computation time. Most of the instances were solved within 1000 seconds. An increasing pattern in objective values can be seen as the size of the instances increases. Unlike BP, ALNS includes a smaller number of passengers on the routes. Across all the test instances, on average, ALNS serves 86% of the passengers, ranging 77%-90%.

Table 6. Parameter Values of ALNS

Parameter	**Value**
Maximum Number of Iteration	10000
η_1	15
η_2	10
η_3	5
Roulette Wheel Parameter	0.7
Shaw Removal Weight κ	0.5
Operator Weight Update Parameter ω	0.99
Number of Passengers to Remove, β	$Uniform(1, \lvert P \rvert/3)$

Table 7. Results of the ALNS algorithm

Instance	No. Drivers	No. Passengers	Execution Time (seconds)	Objective Value	Average of Number Passengers Served
00dc5c5	5	5	358.70	75.99	4.44
01dc5c10	5	10	16.18	163.28	8.93
02dc10c10	10	10	651.42	117.80	9.00
03dc10c15	10	15	1229.99	181.38	13.51
04dc15c15	15	15	29.35	513.00	12.60
05dc15c20	15	20	1705.14	216.22	17.95
06dc20c20	20	20	5640.78	207.80	17.92
07dc20c25	20	25	40.67	414.32	22.51
08dc25c25	25	25	45.98	308.81	22.39
09dc25c30	25	30	58.17	1223.58	24.53
10dc30c30	30	30	382.59	1924.11	22.96
11dc30c35	30	35	68.14	1982.41	27.44
12dc35c35	35	35	76.18	1276.38	29.96
13dc35c40	35	40	559.07	1139.49	35.16
14dc40c40	40	40	578.07	22619.01	31.20
15dc40c45	40	45	706.55	61249.25	40.29
16dc45c45	45	45	737.41	29246.39	37.33
17dc45c50	45	50	722.78	1828.32	42.29
18dc50c50	50	50	775.80	35276.91	39.96

Comparison with the Wait-and-See Solution

In this section, we analyze how our proposed methods compare to a solution where we have perfect information about the uncertainties (i.e., true optimal). This is called the Wait-and-See (WS) solution, and it is determined with the following objective function:

$$\mathrm{E}_{\xi\in\Xi}[\min_{x,\alpha} \sum_{\mathrm{i}\in\mathrm{N}} \sum_{j\in N} \sum_{k\in D} \alpha_{i,j} x_{i,j,v} + \mathbb{Q}(x,\alpha,\xi)]$$

In other words, we solve the problem for each sample and take expectation over all the objective values. For this purpose, we generate 15 test samples. These samples are generated

using a fixed seed number to control the randomness. We then solve the stochastic problem using BP and ALNS and evaluate the solution on those same 15 test samples. We report the WS objective values and the optimality gap of our two proposed methods. The results are shown in Table 8.

The results show that the Branch and Price method provides a solution within 23% of the WS solution on average. ALNS provides a solution that is within 28% of the WS solution on average. In four of the five instances tested, BP outperforms ALNS in terms of solution quality. In one instance, however, ALNS gives a solution closer to the WS solution than BP. From the discussion, we can conclude that although BP generally gives better solutions, its high computation time prohibits it from solving larger instances. ALNS, on the other hand, provides a worse solution but can solve much larger instances with a limited computational time budget.

Table 8. Comparison with Wait-and-See Solution

			WS	Branch and Price		ALNS	
Instance	**No. Drivers**	**No. Passengers**	**Objective**	**Objective**	**% gap**	**Objective**	**% gap**
00dc5c5	5	5	58.82	70.99	17%	78.93	25%
01dc5c10	5	10	160.89	207.82	23%	208.80	23%
02dc10c10	10	10	77.19	99.10	22%	130.07	41%
03dc10c15	10	15	130.36	152.41	14%	136.43	4%
04dc15c15	15	15	82.11	128.91	36%	155.08	47%

Conclusion

In this project, we proposed two solution methods for solving the rideshare routing problem with flexible meeting points under uncertainty. We considered two sources of uncertainties: travel time and passenger cancellation. We integrated both sources of uncertainties into a two-stage stochastic program. We later decomposed the model into a separate two-stage stochastic rideshare routing problem and a quadratic meeting point selection problem.

We then present two of our solution approaches. Our first approach uses Branch and Price to solve the rideshare routing problem. To achieve this, we decompose the rideshare routing problem into a master problem and pricing subproblem. The master problem selects the best set of routes from a subset of feasible routes. The sub-problem generates new feasible routes as needed. We use Sample Average Approximation to iteratively generate samples and solve the problem repeatedly until the optimality gap is sufficiently small.

Our next approach uses Adaptive Large Neighborhood Search (ALNS) to solve the rideshare routing problem. We use five selection operators and four perturbation operators. We use a scoring system to monitor the performance of each of the operators and use a roulette wheel

mechanism to select the operators. We use a maximum iteration number and a convergence rate parameter as the stopping criteria. As before, we use SAA to iteratively sample and resolve the problem until the SAA gap becomes sufficiently small.

We also propose a data-driven approach where, instead of assuming a distribution for travel time, we use Kernel Density Estimate (KDE) to learn the underlying distribution and generate new samples. We also use KDE to generate new samples about passenger cancellation.

Results of the numerical experiments and comparison with the Wait-and-See solution show that the Branch-and-Price approach provides a more accurate solution under uncertainty. ALNS, on the other hand, can solve large problem instances within reasonable computation time.

The methods proposed in this project can provide transportation officials and ridesharing planners insights into how to incorporate uncertainty into rideshare routing where uncertainty distribution may not be known. Since uncertainty is inherent in transportation systems, our proposed problem and solution methods can help planners provide routes to rideshare drivers that are more robust under uncertainty compared to deterministic routes.

References

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., & Vance, P. H. (1998). Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research*, *46*(3), 316–329. https://doi.org/10.1287/opre.46.3.316

Che, Y., & Zhang, Z. (2023). An Integer L-shaped algorithm for vehicle routing problem with simultaneous delivery and stochastic pickup. *Computers & Operations Research*, *154*, 106201. https://doi.org/10.1016/j.cor.2023.106201

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

Dessouky, M. (2022). *The Ridesharing Routing Problem with Flexible Pickup and Drop-off Points*. https://doi.org/10.7922/G2M32T3Z

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

Hou, L., & Zhou, H. (2010). Stochastic Vehicle Routing Problem with Uncertain Demand and Travel Time and Simultaneous Pickups and Deliveries. *2010 Third International Joint Conference on Computational Science and Optimization*, 32–35. https://doi.org/10.1109/CSO.2010.38

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

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, 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

Long, J., Tan, W., Szeto, W. Y., & Li, Y. (2018). Ride-sharing with travel time uncertainty. *Transportation Research Part B: Methodological*, *118*, 143–171. https://doi.org/10.1016/j.trb.2018.10.004

Lu, C., Wu, Y., & Yu, S. (2022). A Sample Average Approximation Approach for the Stochastic Dial-A-Ride Problem on a Multigraph with User Satisfaction. *European Journal of Operational Research*, *302*(3), 1031–1044. https://doi.org/10.1016/j.ejor.2022.01.033

New York (N.Y.). Taxi And Limousine Commission. (2019). *New York City Taxi Trip Data, 2009-2018: Version 1* (Version v1) [dataset]. ICPSR - Interuniversity Consortium for Political and Social Research. https://doi.org/10.3886/ICPSR37254.V1

Oyola, J., Arntzen, H., & Woodruff, D. L. (2017). The stochastic vehicle routing problem, a literature review, Part II: Solution methods. *EURO Journal on Transportation and Logistics*, *6*(4), 349–388. https://doi.org/10.1007/s13676-016-0099-7

Oyola, J., Arntzen, H., & Woodruff, D. L. (2018). The stochastic vehicle routing problem, a literature review, part I: Models. *EURO Journal on Transportation and Logistics*, *7*(3), 193–221. https://doi.org/10.1007/s13676-016-0100-5

Ropke, S., & Pisinger, D. (2006). An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, *40*(4), 455–472. https://doi.org/10.1287/trsc.1050.0135

Saint-Guillain, M., Solnon, C., & Deville, Y. (2017). The Static and Stochastic VRP with Time Windows and both Random Customers and Reveal Times. In G. Squillero & K. Sim (Eds.), *Applications of Evolutionary Computation* (Vol. 10200, pp. 110–127). Springer International Publishing. https://doi.org/10.1007/978-3-319-55792-2_8

Shaw, P. (1998). Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In M. Maher & J.-F. Puget (Eds.), *Principles and Practice of Constraint Programming—CP98* (Vol. 1520, pp. 417–431). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-49481-2_30

Sungur, I., Ren, Y., Ordóñez, F., Dessouky, M., & Zhong, H. (2010). A Model and Algorithm for the Courier Delivery Problem with Uncertainty. *Transportation Science*, *44*(2), 193–205. https://doi.org/10.1287/trsc.1090.0303

Taş, D., Gendreau, M., Dellaert, N., Van Woensel, T., & De Kok, A. G. (2014). Vehicle routing with soft time windows and stochastic travel times: A column generation and branch-and-price solution approach. *European Journal of Operational Research*, *236*(3), 789–799. https://doi.org/10.1016/j.ejor.2013.05.024

Wang, Z., Dessouky, M., Van Woensel, T., & Ioannou, P. (2023). Pickup and delivery problem with hard time windows considering stochastic and time-dependent travel times. *EURO Journal on Transportation and Logistics*, *12*, 100099. https://doi.org/10.1016/j.ejtl.2022.100099

Zhang, W., Jacquillat, A., Wang, K., & Wang, S. (2023). Routing Optimization with Vehicle–Customer Coordination. *Management Science*, *69*(11), 6876–6897. https://doi.org/10.1287/mnsc.2023.4739

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. The New York Taxicab Dataset is found in PARQUET format.

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 figshare: https://doi.org/10.6084/m9.figshare.24208827. This includes the travel times and passenger cancellation historical data, distance matrix, coordinates of the zones and, all the data in the tables, and the data for the graphs. The New York Taxicab and Limousine Commission Dataset can be found at: https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page

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.