# Ride-Sharing Optimization Algorithms for Urban Commuting

by

Mohd. Hafiz Hasan

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2021

Doctoral Committee:

Professor Pascal Van Hentenryck, Co-Chair
Professor Seth Pettie, Co-Chair
Professor John E. Laird
Associate Professor Viswanath Nagarajan

Mohd. Hafiz Hasan

hasanm@umich.edu

ORCID iD: 0000-0002-1930-822X

To my wife Zurita and my children Harris, Hakeem, and Hannah without whom I could not have completed this journey

# ACKNOWLEDGMENTS

I am very blessed to have made it this far in this Ph.D. program, and I would not have been able to have it not been for the following fine people. I am most indebted to my research advisor, Pascal Van Hentenryck, whose patience and dedication—always making time for advice and guidance no matter how busy he was—has taught me what it meant to be a true research scientist, and for his never-ending support, without which this dissertation would have never seen the light of day. My deepest gratitude goes out to my academic advisors, Grant Schoenebeck and Seth Pettie, for always being available for advice and guidance and for helping me navigate this Ph.D. program. I am also extremely grateful to my wife, Zurita Azrina Ayob, my children, Harris Mohd. Hafiz, Hakeem Mohd. Hafiz, and Hannah Mohd. Hafiz, and my parents, Hasan Ahmad and Hasleena Abdullah, for being my unwavering pillars of support throughout this entire journey. I would also like to thank Antoine Legrain for being a trusted mentor and confidant, Ceren Budak, Viswanath Nagarajan, and John Laird for providing feedback for my work, Steve Dolen for providing the dataset that was instrumental to this research, Karen Liska, Cathy Boblitt, Ashley Andreae, and Lisa Cox for their professionalism and prompt response to any of my administrative concerns, the EECS 280 staff, especially James Juett, for becoming my surrogate computer science family, and my colleagues from the Van Hentenryck Lab: Connor Riley, Terrence Mak, Ferdinando Fioretto, Geunyeong Byeon, Xilei Zhao, Jorge Huertas, Joyce Chen, Benjamin Reeves, and Jacob Ketterer for their camaraderie. Last but not least, I would like to acknowledge the Rackham Graduate Student Research Grant for providing the financial resources that were critical to the completion of this research and Advance Research Computing at the University of Michigan for furnishing the computational resources that were used throughout this research.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF APPENDICES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| **ATSP** | Asymmetric Traveling Salesman Problem |
| **ATSPTW** | Asymmetric Traveling Salesman Problem with Time Windows |
| **AV** | autonomous vehicle |
| **BPA** | Branch-and-Price Algorithm |
| **CPP** | Car-Pooling Problem |
| **CTSP** | Commute Trip-Sharing Problem |
| **CTSPAV** | Commute Trip-Sharing Problem for Autonomous Vehicles |
| **DARP** | Dial-a-Ride Problem |
| **DFJ** | Dantzig-Fulkerson-Johnson |
| **DTI** | delivery triangle inequality |
| **ESPPRC** | Elementary Shortest Path Problem with Resource Constraints |
| **FCTSP** | Flexible Commute Trip-Sharing Problem |
| **GPS** | global positioning system |
| **LP** | linear-programming |
| **LTP** | Logistics, Transportation, and Parking |
| **MIDAS** | Michigan Institute for Data Science |
| **MIP** | mixed-integer program |
| **MP** | master problem |
| **MREA** | Mini Route-Enumeration Algorithm |
| **MTZ** | Miller-Tucker-Zemlin |
| **NYC** | New York City |

**PCATSP** Precedence-Constrained Asymmetric Traveling Salesman Problem

**PDP** Pickup and Delivery Problem

**PDPTW** Pickup and Delivery Problem with Time Windows

**PSP** pricing subproblem

**QoS** quality of service

**RCSPA** Resource-Constrained Shortest Path Algorithm

**REA** Route-Enumeration Algorithm

**RMP** restricted master problem

**RT-CTSP** Real-Time Commute Trip-Sharing Problem

**SAV** shared autonomous vehicle

**SCC** strongly-connected component

**SEC** subtour-elimination constraint

**SMS** shared mobility services

**SPPRC** Shortest Path Problem with Resource Constraints

**TLC** Taxi and Limousine Commission

**TSP** Traveling Salesman Problem

**VC** vehicle count

**VMT** vehicle miles traveled

**VRP** Vehicle Routing Problem

**VRPTW** Vehicle Routing Problem with Time Windows

# ABSTRACT

As cities struggle to cope with the ever-increasing demand on their transportation infrastructures, ride-hailing services have emerged as a potential remedy that promises to revolutionize urban mobility by making on-demand transportation available at the touch of a fingertip. The long-term sustainability of these services, however, can only be realized when their rides are aggregated by having individual vehicles serve multiple trips simultaneously to maximize the utilization of available seat capacity, i.e., by "true" ride sharing. While the community has long recognized ride sharing's potential for reducing traffic congestion, energy consumption, parking utilization, and greenhouse gas emissions, numerous unsolved challenges—from providing attractive mechanisms to incentivize modal shifts to building trust among unacquainted passengers—remain to hinder its widespread adoption.

A key obstacle to ride sharing's ubiquity is the difficulty of coordinating rides with matching locations and schedules combined with the absence of algorithms capable of matching riders and drivers quickly and effectively. This research addresses this challenge by focusing on finding optimal routing plans for fleets of conventional and autonomous vehicles that maximize ride sharing for commute trips to power future ride-sharing platforms. The need to design routes that match trips to and from the workplace—that in turn, are dispersed spatially and temporally with schedules that may change every day—while respecting time-window, ride-duration, and vehicle-capacity constraints highlights the complexity of the problem. Driven by an original desire to investigate the potential of optimized ride-sharing platforms in relieving the parking pressure induced by the thousands of commuters traveling to the University of Michigan campus in Ann Arbor, Michigan, this

research: (1) develops the mathematical framework for modeling the problem of seeking the optimal routing plan that maximizes ride sharing for commute trips for conventional and autonomous vehicles, (2) proposes techniques to decompose the problem and designs exact and approximate algorithms to tackle its computational complexity, and (3) quantifies the potential benefits and drawbacks of the generated plans and provides insight into the different factors that influence their performance through a real case study.

Aside from investigating modeling and decomposition techniques that specifically exploit the structure imposed by the problem constraints and the spatio-temporal characteristics of the trips, this research also proposes solution approaches that leverage state-of-the-art linear-programming and combinatorial-optimization techniques, ranging from column generation to discover useful routes on demand to dynamic programming to efficiently find resource-constrained least-cost paths. The solution approaches share a common characteristic: Each produces a valid lower bound to the objective value which allows the calculation of an optimality gap to quantify its solution quality. These algorithms are further bolstered by the availability of a real-world dataset of the commute trips made by 15,000 drivers that use 15 university-operated parking structures in downtown Ann Arbor over April 2017; it not only allows the algorithms to be evaluated on real-world data, but analyses of its results provide invaluable insights into the performance characteristics of the optimized routing plans. This research demonstrates that through the optimal plans, the number of vehicles for these trips can be potentially reduced by 57% and 92% when using conventional and autonomous vehicles respectively. It also quantifies numerous other potential benefits and drawbacks from utilizing the plans, some of which include reductions in vehicle usage during peak hours, decreases in vehicle miles traveled, and increases in average commute times.

# CHAPTER 1

# Introduction

The emergence of smartphones and ride-hailing services have revolutionized the landscape of urban mobility by making on-demand transportation available at the touch of a fingertip; however, traffic congestion remains a growing burden on urban areas as vehicle ownership continues to rise. The INRIX 2017 Traffic Scorecard (Cookson 2018) estimates that traffic congestion has costed the US economy more than $305 billion in 2017 alone, which is up $10 billion from the previous year. While car-pooling services provide an appealing alternative due to their potential benefits, be it in reducing traffic congestion, energy consumption, or parking utilization (Shaheen and Rodier 2005), their adoption remains poor. In fact, solo driving has remained as the overwhelming choice for daily commuting (McKenzie 2015) due to several challenges associated with car pooling. A study on factors influencing car-pool formation by Li et al. (2007) revealed difficulty in finding people with matching locations and schedules as the primary barrier to car pooling. This highlights the opportunity available for matching platforms to alleviate this burden by automatically identifying commuting groups based on factors that are consequential to individuals' commuting decisions. It also emphasizes the critical role of rider-matching and route-optimization algorithms to enable such platforms.

To this end, this dissertation considers the problem of *designing optimal routing plans for fleets of conventional and autonomous vehicles that maximize ride sharing for commute trips* to power these platforms. It envisions a platform that utilizes a reservation-based system to receive (recurring) commute-trip requests ahead of time and then generates a routing plan that optimally matches riders (and drivers) to maximize ride sharing. The complexity of the optimization problem stems from the need to match trips to and from the workplace that are dispersed spatially and temporally and whose schedules may change every day. A successful ride-sharing platform for commuting would necessarily require efficient solution approaches that are capable of solving the problem quickly, yet very little attention has been given to their development. This work aims to fill this void by *synthesizing mathe-*

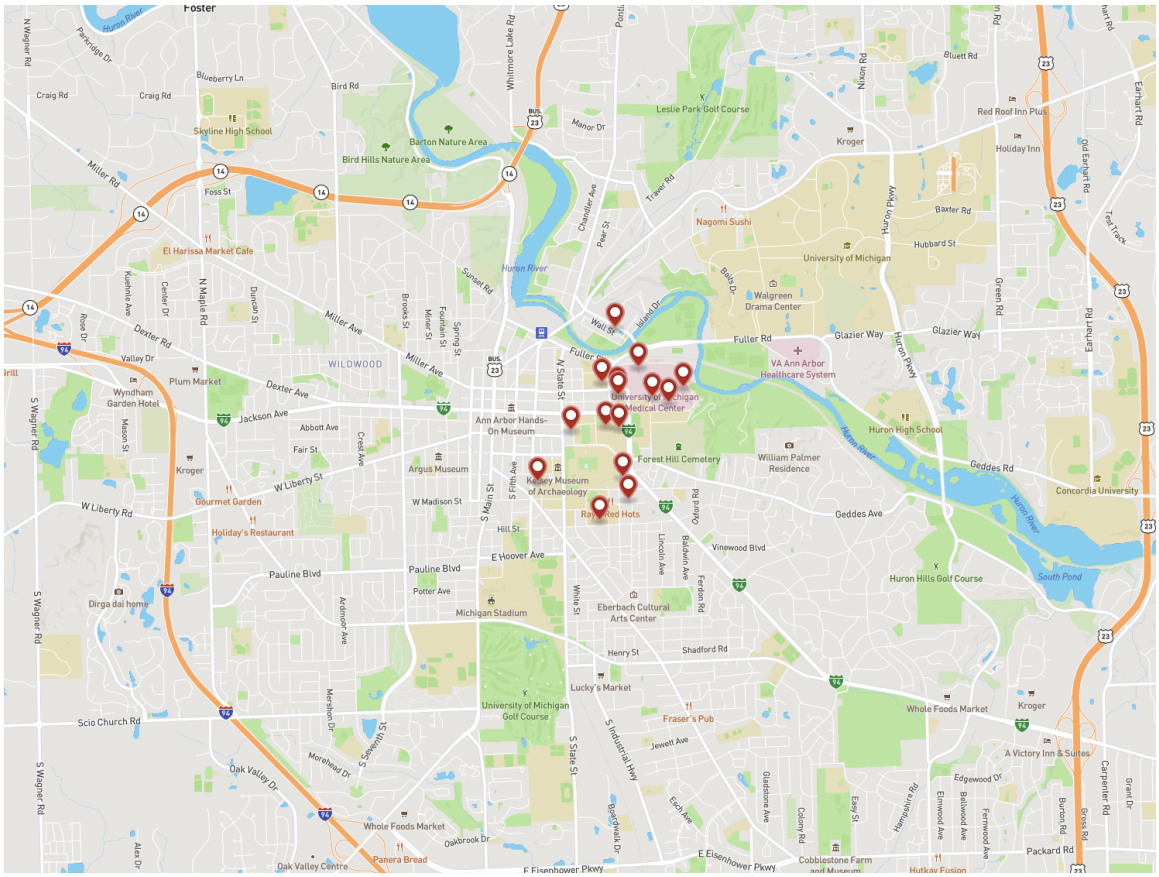



Figure 1.1: Locations of the 15 Parking Structures Operated by the University of Michigan in Downtown Ann Arbor.

*matical models of the problem and developing decomposition and algorithmic techniques to effectively solve them.*

This study was originally motivated by a desire to investigate the potential of optimized ride-sharing platforms in relieving the parking pressure induced by the thousands of daily commuters traveling to the University of Michigan campus in Ann Arbor, Michigan. Being the largest employer in the city with more than 50,000 employees, the university operates several large parking structures located in the downtown area which are not only expensive but are also located at prime locations for the convenience of the commuters. If not for their high demand, these prime spaces—that are utilized to park idle vehicles during the day and that are mostly vacant during the weekends—could otherwise be used to house other infrastructures that could bring more economical or recreational benefit to the surrounding community. A collaboration between the Michigan Institute for Data Science (MIDAS) and the Logistics, Transportation, and Parking (LTP) division of the university allowed us to gather detailed information about the commuting patterns of approximately 15,000 drivers who used 15 of these university-operated parking structures over the month of April

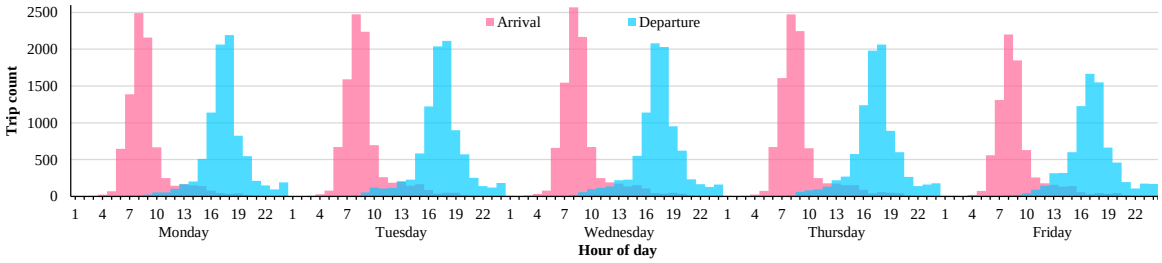



Figure 1.2: Arrival and Departure Time Distributions on Second Week of April 2017.

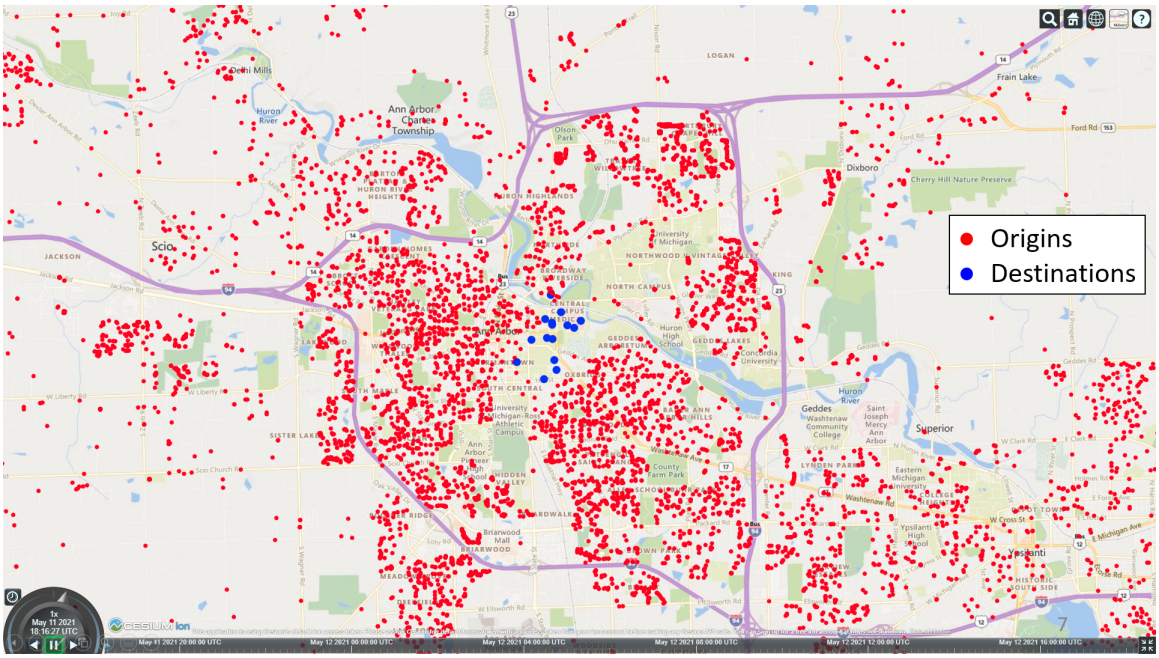



Figure 1.3: Commuting Origins and Destinations from Ann Arbor Commute-Trip Dataset.

2017 (the locations of these structures are displayed in Figure 1.1). The data consisted of the precise arrival and departure times of these commuters to and from the structures as well as their home addresses, which are located within an area spanning 13,000 square miles that covers the city of Ann Arbor as well as its surrounding neighborhoods. Figure 1.2 shows the distribution of the arrival and departure times of this population to and from the parking structures over the busiest week of the month. Their travel patterns display a remarkable amount of consistency and similarity, with the peak arrival and departure times coinciding with the typical 6–9 am and 4–7 pm peak commute hours respectively every day. Next, Figure 1.3 provides an overview of the spatial distribution of the commuting origins (home locations) of the population relative to the destinations (parking structures) obtained by geocoding every address into their global positioning system (GPS) coordinates.

The availability of this rich dataset allowed us to reconstruct the daily trips of these

commuters, which then not only permitted us to evaluate our proposed solution approaches on real-world data, but also enabled us to further analyze the computational results to unearth key insights into the benefits and drawbacks of these ride-sharing platforms and consequentially reveal the factors that are critical to the platforms' performance. The key contributions of this dissertation can be summarized as follows:

1. It develops the mathematical framework for modeling the problem—which seeks the optimal routing plan that maximizes ride sharing for commute trips—for conventional and autonomous vehicles.
2. It proposes techniques to decompose the problem and designs exact and approximate algorithms to address its computational complexity.
3. It quantifies the potential benefits and drawbacks of the generated plans and provides insight into the different factors that influence their performance through a real case study.

The rest of this chapter is organized as follows. Section 1.1 first outlines the key concepts and terminologies that are used throughout this dissertation, while Section 1.2 reviews the main optimization techniques that are used in the dissertation. Section 1.3 then reviews literature related to this study. Finally, Section 1.4 provides an overview of this dissertation: It describes how the dissertation is structured and summarizes the contents of each chapter.

## 1.1 Concepts and Terminologies

This section outlines the key concepts and terminologies that are used throughout this dissertation: commute trips, characteristics of routes of conventional and autonomous vehicles (and their key differences), and optimal routing plans. This work assumes the utilization of a homogeneous fleet of vehicles with capacity $K$ to serve all rides, and that the *triangle inequality* is satisfied for all travel times and distances.

### 1.1.1 Commute Trips

A trip $t = \{o, d, dt, at\}$ is a tuple that consists of an origin–destination pair, $o$ and $d$, which specifies the pickup and drop-off locations of the trip, and the desired service times at both locations; a departure time $dt$ at the origin and an arrival time $at$ at the destination. On any day, a commuter $c$ makes two trips: a trip to the workplace, $t_c^+$, and a trip back home, $t_c^-$. These trips are referred to henceforth as inbound and outbound trips respectively. A round trip, $t_c^{rt} = (t_c^+, t_c^-)$, is the pair of inbound and outbound trips made by commuter $c$.

### 1.1.2 Routes of Conventional Vehicles

A route $r_{\mathcal{T}}$ is a sequence of origin and destination locations from a set of inbound or outbound trips, $\mathcal{T}$, whereby each origin and destination from the set is visited *exactly once* (i.e., it is elementary). A route that visits the origin $o_c$ of a trip $t_c$ must also visit its destination $d_c$ (and vice versa), and it must visit the origin before the destination. These are referred to as the *pairing and precedence constraints* respectively. For instance, a possible route for trips $t_1 = \{o_1, d_1, dt_1, at_1\}$ and $t_2 = \{o_2, d_2, dt_2, at_2\}$ is $r_{\{t_1,t_2\}} = o_2 \rightarrow o_1 \rightarrow d_1 \rightarrow d_2$. An inbound route $r^+$ covers only inbound trips and an outbound route $r^-$ covers only outbound trips. Each route $r$ serves a set of riders $\mathcal{C}_r$. For a conventional vehicle, the driver $D_r$ of a route $r$ comes from its set of riders, i.e., $D_r \in \mathcal{C}_r$. The route must therefore begin at its driver's origin and end at her destination. These are referred to as the *driver constraints*. For instance, rider 2 must be the driver of route $o_2 \rightarrow o_1 \rightarrow d_1 \rightarrow d_2$. The total number of riders on the vehicle at any point along a route cannot exceed its capacity. In other words, a route must satisfy the *vehicle-capacity constraints*.

**Definition 1.1.1** (Valid Route)**.** A valid route $r$ visits $o_c$ before $d_c$ for every rider $c \in \mathcal{C}_r$, starts at $o_{D_r}$ and ends at $d_{D_r}$, respects the vehicle capacity, and is elementary.

Let $T_i$ denote the time at which service begins at location $i$, $\zeta_i$ be the service duration at $i$, $pred(i)$ denote the location on a route visited just before $i$, and $\tau_{(i,j)}$ be the estimated travel time for the shortest path between locations $i$ and $j$. It is assumed that commuters sharing rides are willing to tolerate some inconvenience in terms of deviations to their trips' desired departure and arrival times as well as in terms of extensions to the ride durations of their individual trips. Therefore, a time window $[a_i, b_i]$ is constructed around the desired times and is associated with each pickup location $i$, where $a_i$ and $b_i$ denote the earliest and latest times at which service may begin at $i$ respectively. Conversely, only an upper bound $b_j$ is associated with each drop-off location $j$ as the arrival time at $j$ is implicitly bounded from below by $a_j = a_i + \zeta_i + \tau_{(i,j)}$, where $i$ is the corresponding pickup location for $j$. Finally, a duration limit $L_t$ is associated with each trip $t$ to denote its maximum ride duration.

**Definition 1.1.2** (Feasible Route)**.** A feasible route $r_{\mathcal{T}}$ is a valid one with pickup and drop-off times $T_i \in [a_i, b_i]$ for each location $i \in r_{\mathcal{T}}$ that ensures the ride duration of each trip $t \in \mathcal{T}$ does not exceed $L_t$.

A feasible route is therefore a valid one that satisfies the *time-window constraints* of its locations as well as the *ride-duration limit constraints* of its trips. Determining if a

valid route $r$ is feasible amounts to solving a feasibility problem defined by the following constraints on $r$:

$$a_{o_c} \leq T_{o_c} \leq b_{o_c} \qquad \forall c \in \mathcal{C}_r \tag{1.1}$$

$$T_{d_c} \leq b_{d_c} \qquad \forall c \in \mathcal{C}_r \tag{1.2}$$

$$T_{pred(o_c)} + \zeta_{pred(o_c)} + \tau_{(pred(o_c),o_c)} \leq T_{o_c} \qquad \forall c \in \mathcal{C}_r \setminus \{D_r\} \tag{1.3}$$

$$T_{pred(d_c)} + \zeta_{pred(d_c)} + \tau_{(pred(d_c),d_c)} = T_{d_c} \qquad \forall c \in \mathcal{C}_r \tag{1.4}$$

$$T_{d_c} - (T_{o_c} + \zeta_{o_c}) \leq L_{t_c} \qquad \forall c \in \mathcal{C}_r \tag{1.5}$$

Constraints (1.1) and (1.2) are the time-window constraints for the pickup and drop-off locations respectively, while constraints (1.3) and (1.4) describe compatibility requirements between pickup/drop-off times and travel times between consecutive locations along the route. Finally, constraints (1.5) specify the ride-duration limit for each rider's trip. Note that constraints (1.3) allow waiting at pickup locations. Moreover, constraints (1.3) and (1.4) enforce strictly increasing start of service times at consecutive locations along $r$, which then ensures that the route is elementary. Numerous algorithms have been proposed for solving this feasibility problem efficiently, e.g. Tang et al. (2010), Haugland and Ho (2010), Firat and Woeginger (2011), and Gschwind and Irnich (2015). In the sequel, the Boolean function $feasible(r)$ is used to indicate whether a route $r$ admits a feasible solution to constraints (1.1)–(1.5).

**Definition 1.1.3** (Feasible Round-Trip Route)**.** Let $r_{\mathcal{T}^+}$ and $r_{\mathcal{T}^-}$ denote feasible routes for a set of inbound trips $\mathcal{T}^+$ and a set of outbound trips $\mathcal{T}^-$ respectively. A feasible round-trip route $r_{\mathcal{T}^{rt}} = (r_{\mathcal{T}^+}, r_{\mathcal{T}^-})$ is a pair of feasible inbound and outbound routes serving the same set of commuters, i.e., $\mathcal{C}_{r_{\mathcal{T}^+}} = \mathcal{C}_{r_{\mathcal{T}^-}}$, and having the same driver, $D_{r_{\mathcal{T}^+}} = D_{r_{\mathcal{T}^-}}$.

### 1.1.3 Routes of Autonomous Vehicles

The key difference between routes of autonomous vehicles (AVs) and conventional vehicles is the absence of drivers from the former. They are therefore not subjected to the driver constraints described in Section 1.1.2; instead, an AV route begins and ends at a depot. Some algorithms in this work attempt to leverage the spatial structure of the origins and destinations from the Ann Arbor dataset by decomposing an AV route into a sequence of shorter constituents called *mini routes*.

**Mini Routes** A mini route $r$ is a sequence of locations that visits each origin and destination from a set of inbound or outbound trips exactly once. A mini route $r$ must respect

the vehicle capacity, i.e., $|\mathcal{C}_r| \leq K$, and consists of three phases: a pickup phase where passengers are picked up, a transit phase where the vehicle travels to the destination, and a drop-off phase where all the passengers are dropped off. During the pickup (resp. drop-off) phase, the vehicle visits only origins (resp. destinations), whereas it travels from an origin to a destination in the transit phase. For instance, a possible mini route for a car with $K = 4$ serving trips $t_1 = \{o_1, dt_1, d_1, at_1\}$, $t_2 = \{o_2, dt_2, d_2, at_2\}$, and $t_3 = \{o_3, dt_3, d_3, at_3\}$ is $r = o_2 \rightarrow o_1 \rightarrow o_3 \rightarrow d_1 \rightarrow d_2 \rightarrow d_3$, and its pickup, transit, and drop-off phases are given by $o_2 \rightarrow o_1 \rightarrow o_3$, $o_3 \rightarrow d_1$, and $d_1 \rightarrow d_2 \rightarrow d_3$ respectively. A mini route travels only in a single direction (either inbound or outbound), therefore, an inbound mini route $r^+$ covers only inbound trips and an outbound mini route $r^-$ covers only outbound trips.

**Definition 1.1.4** (Valid Mini Route)**.** A valid mini route $r$ serving a set $\mathcal{C}_r$ of riders visits all of its origins, $\{o_c : c \in \mathcal{C}_r\}$, before its destinations, $\{d_c : c \in \mathcal{C}_r\}$, covers only inbound or outbound trips, respects the vehicle capacity, i.e., it has $|\mathcal{C}_r| \leq K$, and is elementary.

This validity requirement bounds the number of locations visited by a mini route $r$ to $2K$. Let $\dot{\mathcal{C}}_r$ denote the first commuter served on $r$. Similar to a conventional-vehicle route, for a valid mini route to be feasible, it has to satisfy the time-window constraints at each location and the ride-duration limit constraints of its trips.

**Definition 1.1.5** (Feasible Mini Route)**.** A feasible mini route $r$ is valid, has pickup and drop-off times $T_i \in [a_i, b_i]$ for each location $i \in r$, and ensures the ride duration of each rider $c \in \mathcal{C}_r$ does not exceed $L_{t_c}$.

Determining if a valid mini route $r$ is feasible amounts to solving a feasibility problem defined by constraints (1.1), (1.2), (1.4), (1.5), and

$$T_{pred(o_c)} + \zeta_{pred(o_c)} + \tau_{(pred(o_c),o_c)} \leq T_{o_c} \qquad \forall c \in \mathcal{C}_r \setminus \dot{\mathcal{C}}_r. \tag{1.6}$$

Constraints (1.6) describe increasing start of service time requirements between pickup locations and their predecessors. They allow waiting at pickup locations, and together with constraints (1.4), they ensure the service starting times at consecutive locations along $r$ are strictly increasing, thus enforcing the elementarity of $r$. The algorithms for checking the feasibility of conventional-vehicle routes listed in Section 1.1.2 are also applicable to this feasibility problem, therefore the Boolean function $feasible(r)$ is also used to indicate whether a mini route $r$ admits a feasible solution to constraints (1.1), (1.2), (1.4), (1.5), and (1.6).

**AV Routes** An AV route $\rho = v_s \rightarrow r_1 \rightarrow \ldots \rightarrow r_k \rightarrow v_t$ is a sequence of $k$ distinct mini routes that starts at a source node $v_s$ and ends at a sink node $v_t$, both representing a designated depot.

**Definition 1.1.6** (Feasible AV Route)**.** A feasible AV route $\rho$ consists of a sequence of distinct, feasible mini routes, starts and ends at a designated depot, and is elementary.

In other words, for $\rho$ to be feasible, each of its mini routes must be valid and satisfy constraints (1.1), (1.2), (1.4), (1.5), and (1.6). Let $\dot{r}$ denote the first location visited on $r$ and $\ddot{r}$ denote the last. Moreover, each mini route $r_i$ $(1 \leq i \leq k)$ must satisfy the following constraints:

$$T_{v_s} + \tau_{(v_s,\dot{r}_1)} = T_{\dot{r}_1} \tag{1.7}$$

$$T_{\ddot{r}_i} + \zeta_{\ddot{r}_i} + \tau_{(\ddot{r}_i,\dot{r}_{i+1})} \leq T_{\dot{r}_{i+1}} \qquad \forall i = 1, \ldots, k-1 \tag{1.8}$$

$$T_{\ddot{r}_k} + \zeta_{\ddot{r}_k} + \tau_{(\ddot{r}_k,v_t)} = T_{v_t} \tag{1.9}$$

Constraints (1.7)–(1.9) describe compatibility requirements between the ending and the beginning service times of consecutive mini routes along $\rho$ and the travel times between them. The constraints, together with (1.4) and (1.6), enforce strictly increasing starting times for service at all consecutive locations along $\rho$, therefore ensuring that $\rho$ is elementary. This work also assumes that an AV may begin and end its route at any time of the day, therefore $T_{v_s}$ and $T_{v_t}$ are not subjected to any time-window constraints.

### 1.1.4 Optimal Routing Plan

A *routing plan* for a set of commuters $\mathcal{C}$ is a *set of feasible routes* (of conventional or AVs) that covers every inbound and outbound trip of $\mathcal{C}$ *exactly once*. An *optimal* routing plan is one that has the *minimum cost*. The problems of finding the optimal routing plans for either conventional or AVs being considered in this study are therefore generalizations of the Vehicle Routing Problem (VRP), a problem which is well known to be NP-hard (Toth and Vigo 2002).

## 1.2 Preliminaries

This section provides an overview of several linear-programming (LP) techniques for solving (large-scale) mixed-integer programs (MIPs) that are used in this dissertation: column generation, branch and price, and branch and cut.

### 1.2.1 Column Generation

Consider the following MIP, referred to as the master problem (MP), that is defined on a large (potentially exponential) set of variables $x = \{x_i : i \in \mathcal{I}\}$:

$$z^*_{\text{MP}} = \min \quad c^\intercal x \tag{1.10}$$

$$\text{s.t.} \quad Ax = b \tag{1.11}$$

$$x_i \in \{0, 1\} \qquad \forall i \in \mathcal{I} \tag{1.12}$$

A potential solution approach to the MP is to first enumerate the columns $\boldsymbol{a}_i$ associated with each $x_i$ for every $i \in \mathcal{I}$ and then solve the MP with a MIP solver. However, the enumeration of $\mathcal{I}$ may not be practical for certain problems if the set is too large.

Column generation is a procedure that solves the LP relaxation of the MP. It obviates the difficulty of enumerating $\mathcal{I}$, especially for problems where the enumeration is impractical, by generating the columns on demand. It is an iterative procedure that repeatedly solves a restricted master problem (RMP) and a pricing subproblem (PSP) until a convergence criterion is satisfied. First, it solves a RMP which is the linear relaxation of the MP that is defined on only a subset of its variables $x' = \{x_i : i \in \mathcal{I}' \subseteq \mathcal{I}\}$, i.e.:

$$z_{\text{RMP}} = \min \quad c'^\intercal x' \tag{1.13}$$

$$\text{s.t.} \quad A'x' = b \tag{1.14}$$

$$x_i \geq 0 \qquad \forall i \in \mathcal{I}' \tag{1.15}$$

It then solves a PSP that searches for new variables that could improve the objective value $z_{\text{RMP}}$, i.e., variables that would enter the basis of the optimal solution to the RMP. Such a variable, $x_e$, would enter the basis if its reduced cost $\bar{c}_e$ is negative, i.e., if

$$\bar{c}_e = c_e - \boldsymbol{\pi}^\intercal \boldsymbol{a}_e < 0, \tag{1.16}$$

where $c_e$ is the objective coefficient of the variable, $\boldsymbol{\pi}$ is the vector of optimal duals of the RMP, and $\boldsymbol{a}_e$ is the column of constraint coefficients of the variable. The entering variable(s) and its associated column(s) is typically identified by solving a combinatorial-optimization problem that exploits the meaning of the variables.

The entering variable(s) is then used to augment the RMP: The augmentation produces a new RMP that is defined on a larger subset of variables $\{x_i : i \in \mathcal{I}' \cup \{e\}\}$. The RMPs and the PSP are solved iteratively, with $\mathcal{I}'$ being progressively enlarged after each iteration, until the PSP cannot identify any new variable with a negative reduced cost. At this point,

$z_{\text{RMP}}$ *converges* to the optimal objective value $z^*$ of the linear relaxation of the original MP. This is because none of the undiscovered variables in $\mathcal{I} \setminus \mathcal{I}'$ would enter the basis if they were available as their reduced costs are non-negative. Therefore, the objective value $z_{\text{RMP}}$ at convergence represents a primal lower bound to $z^*_{\text{MP}}$, and the solution of the RMP constitutes the optimal solution to the MP *if it is integral*.

### 1.2.2 Branch and Price

As mentioned in the previous section, the solution of the RMP at convergence is also optimal for the original MP if it is integral. Otherwise, several approaches can be adopted to yield an integer solution. The first is to simply solve the final RMP with integer variables, i.e., to solve the following problem:

$$z_{\text{RMP-I}} = \min \quad c'^{\intercal} x' \tag{1.17}$$

$$\text{s.t.} \quad A'x' = b \tag{1.18}$$

$$x_i \in \{0, 1\} \qquad \forall i \in \mathcal{I}' \tag{1.19}$$

with a MIP solver. However, the solution to this problem may not be optimal for the original MP as it does not consider the complete set of variables $\{x_i : i \in \mathcal{I}\}$. In fact, its objective value $z_{\text{RMP-I}}$ only constitutes an upper bound to $z^*_{\text{MP}}$, and since $z_{\text{RMP}} = z^* \leq z^*_{\text{MP}} \leq z_{\text{RMP-I}}$, an optimality gap given by $(z_{\text{RMP-I}} - z_{\text{RMP}})/z_{\text{RMP-I}}$ can be calculated to assess the quality of the solution.

An exact approach for obtaining the optimal solution to the MP using column generation is called branch and price: It supplements the classical, LP-based branch-and-bound procedure for solving MIPs by incorporating column generation into its LP bounding phase. The classical branch-and-bound procedure begins by first solving the LP relaxation of the MIP; if all the relaxed variables happen to have integer values in the LP solution, then the solution is also valid and optimal for the original MIP. Otherwise, the procedure selects a variable whose value must be integer but is fractional in the LP relaxation and *branches* on the variable. Suppose such a variable is $x_b$ and its fractional value is $x^*_b$. The branching on $x_b$ is performed by creating two new MIPs that both exclude the fractional value: The first imposes an additional constraint $x_b \leq \lfloor x^*_b \rfloor$, while the second introduces constraint $x_b \geq \lceil x^*_b \rceil$. The added constraints are referred to as the *branching decisions*. The procedure is then repeated on the two MIPs, whereby their LP relaxations are solved and new branching variables are selected and branched on if necessary. This repetition constitutes the exploration of a *search tree*, whereby each new MIP represents a *tree node*.

When the LP relaxation at a tree node produces an integer solution, the solution is designated as the *incumbent* (the best integer solution discovered so far) if one has not been discovered or if its objective value is better than the existing incumbent's. The node is also designated as being *fathomed*, i.e., it becomes a permanent leaf and is not further branched upon. A node is also fathomed when its LP relaxation is infeasible or when the relaxation's objective value is worse than that of the incumbent. The fathoming of the latter prunes the search tree as the node obviously cannot yield an integer solution that is better than the incumbent.

The search procedure also naturally maintains an upper and a lower bound to the optimal objective value. Suppose the original MIP is a minimization problem, then the objective value of the incumbent constitutes the upper bound, whereas the *best bound*, obtained by taking the minimum of the LP-relaxation objective values of all the unexplored nodes (nodes that have yet to be branched upon), constitutes the lower bound. The latter explains why solving the LP relaxation at each node is termed the *bounding phase*. The difference between the upper and the lower bound yields an (absolute) optimality gap, which when zero proves the optimality of the incumbent solution.

The branch-and-price approach is especially useful for finding the exact solution to MIPs whose variable and column sets are too large to be practically enumerated; it addresses the issue by searching for the columns on demand within the classical branch-and-bound framework. More specifically, it utilizes column generation to solve the LP relaxation at each tree node; thus, it may introduce new columns in every bounding phase. Furthermore, the incorporation of column generation makes it necessary to enforce the branching decisions in both the RMP and the PSP to ensure that the bounding phase produces the correct LP bounds.

### 1.2.3 Branch and Cut

The branch-and-cut approach augments the LP-based branch-and-bound procedure for solving MIPs by incorporating *cutting-plane* generation to progressively tighten the LP relaxations. More specifically, after the evaluation of the LP relaxation at each tree node, the approach executes *separation algorithms / heuristics* to identify violated *valid inequalities*: constraints that are satisfied by all feasible integer solutions but are violated by the fractional LP solution. A classical example of such an inequality is Gomory's cut (Gomory 1960). The separated inequalities are then added to the problem formulation to cut off the fractional solution, with the hope of making subsequent LP relaxations "less fractional". As the cutting-plane separation procedure is executed in every bounding phase, the

branch-and-cut approach accumulates the valid inequalities that are violated by previous LP relaxations, cutting off their fractional solutions from all subsequent LP evaluations, and progressively strengthens the best LP bound. This strengthening, in turn, is hoped to expedite the discovery of an optimal solution by further narrowing the optimality gap.

## 1.3 Related Work

The Vehicle Routing Problem with Time Windows (VRPTW) is perhaps the most well-studied variant of the VRP: It seeks a set of minimum cost routes, each departing from and returning to a designated depot, that serve a set of customers, each with a capacity demand and a time window within which service may commence. The problem must ensure that each customer is served exactly once within her time window while not allowing the vehicle capacity to be exceeded. It is well known to be NP-hard, as Savelsbergh (1985) showed that finding a feasible solution to the problem for a fixed vehicle count is strongly NP-complete. It has been studied extensively in the literature, and numerous methods have been suggested to tackle its complexity, from metaheuristics like Taillard et al. (1997) and Bräysy and Gendreau (2005), to exact solution methods using Lagrangian relaxation (Kohl and Madsen 1997, Kallehauge et al. 2006), column generation (Desrosiers et al. 1984, Desrochers et al. 1992), or polyhedral approaches (Kohl et al. 1999, Bard et al. 2002, Kallehauge et al. 2007). An extensive review of the problem is provided by Cordeau et al. (2002).

The VRPTW was generalized to the Pickup and Delivery Problem with Time Windows (PDPTW) by Dumas et al. (1991), whereby service locations come in pairs, a pickup and a delivery location for each customer, that must be serviced in order by the same route. The problem models services that first pick up and then deliver merchandise within specified time windows. They proposed a dynamic-programming, label-setting algorithm to search for routes that satisfy the new pairing and precedence constraints along with the existing time-window and vehicle-capacity constraints. The algorithm is incorporated within a column-generation procedure to solve the problem. A similar approach was also adopted by Ropke and Cordeau (2009) in their branch-and-cut-and-price algorithm for the PDPTW, while Ruland and Rodin (1997) used a polyhedral approach to solve the version of the problem without time-window constraints. The Dial-a-Ride Problem (DARP) generalizes the PDPTW by introducing ride-duration constraints which limit the maximum duration between each pickup and delivery location pair. The constraints model the maximum time spent on the vehicle by every customer and is critical for guaranteeing a quality of service (QoS) for services that transport passengers instead of merchandise, like door-to-door

transportation services for the disabled or the elderly or those for ride sharing. The problem has also been extensively reviewed by Cordeau and Laporte (2003a, 2007), and it has been tackled with methods ranging from approximate ones like heuristics (Bodin and Sexton 1986, Jaw et al. 1986) and metaheuristics (Cordeau and Laporte 2003b, Ritzinger et al. 2016), to exact ones utilizing cutting-plane methods (Cordeau 2006) or column generation (Gschwind and Irnich 2015).

Of the many solution approaches proposed for the various generalizations of the VRP, column generation is perhaps the most popular due to its elegance in only considering a subset of the feasible routes that can improve the objective function, and its proven effectiveness in producing strong lower bounds to the problem objective when used in conjunction with the Dantzig–Wolfe decomposition (Dantzig and Wolfe 1960). The latter technique decomposes an edge-flow formulation of the VRP into a master problem and a pricing subproblem. The master problem typically solves a set-partitioning/covering problem on a set of feasible routes to ensure every customer is served, whereas the pricing subproblem searches for new feasible routes to be added to the set. The latter problem uses the duals of the linear relaxation of the master problem to identify new routes satisfying problem-specific feasibility constraints with negative reduced costs. It is typically cast as an Elementary Shortest Path Problem with Resource Constraints (ESPPRC), whereby resource contraints are used to model the feasibility constraints, and the elementarity requirement ensures that each customer is serviced exactly once. Unfortunately, the ESPPRC has been proven to be NP-hard in the strong sense by Dror (1994), and while exact solution methods have been proposed for the problem, e.g., Feillet et al. (2004), Chabrier (2006), Boland et al. (2006), and Drexl (2013), the elementarity requirement is often relaxed to produce a Shortest Path Problem with Resource Constraints (SPPRC) which admits a pseudo-polynomial solution approach. A variety of strategies are then adopted to handle non-elementary paths, e.g., Desrosiers et al. (1984), Dumas et al. (1991), Ropke and Cordeau (2009), and Gschwind and Irnich (2015) eliminate them by either preventing their selection in an integer solution or by using infeasible-path elimination constraints in the master problem, while Desrochers et al. (1992) and Irnich and Villeneuve (2006) take a middle-ground approach by eliminating 2- and $k$-cycles from the discovered paths respectively. Regardless of whether an SPPRC or an ESPPRC is used in the pricing subproblem, they are typically solved via dynamic-programming algorithms, the most popular being the generalized label-setting algorithm for multiple resource constraints by Desrochers (1988). Other suggested dynamic-programming approaches include the label-correcting algorithm by Desrosiers et al. (1983) which is based on the Ford-Bellman-Moore algorithm and the label-setting algorithm by Desrochers and Soumis (1988) which generalizes

Dijkstra's algorithm. Methods utilizing Lagrangian relaxation (Beasley and Christofides 1989, Borndörfer et al. 2001), constraint programming (Rousseau et al. 2004), heuristics (Desaulniers et al. 2008), and cutting planes (Drexl 2013) have also been proposed. An in-depth overview of the SPPRC is provided by Irnich and Desaulniers (2005).

Another common approach to solving routing problems is the polyhedral approach which generates cutting planes to progressively "trim" the convex hull defining the feasible region of the problem's linear relaxation. Its application on VRPs traces its roots back to the seminal work by Dantzig et al. (1954) for solving the Traveling Salesman Problem (TSP). Their procedure uses an edge-flow formulation of the problem which is iteratively solved to identify subtours which break the feasibility of the solution. A family of valid inequalities, commonly referred to now as the Dantzig-Fulkerson-Johnson (DFJ) subtour-elimination constraints (SECs), are then progressively introduced to prevent generation of the subtours in subsequent solutions. Grötschel and Padberg (1975) later proved that the DFJ SECs induce facets of the polytope of the convex hull of the feasible solutions, which explains why they were so effective at strengthening the LP bound, while Padberg and Rinaldi (1990) proposed an exact algorithm for separating the inequalities. In a similar vein, many other works have focused on identifying facet-defining inequalities together with algorithms/heuristics for separating the violated inequalities, e.g., $D_k^+$ and $D_k^-$ inequalities for the TSP by Grötschel and Padberg (1985), predecessor and successor inequalities for the Precedence-Constrained Asymmetric Traveling Salesman Problem (PCATSP) by Balas et al. (1995), tournament and generalized tournament constraints for the Asymmetric Traveling Salesman Problem with Time Windows (ATSPTW) by Ascheuer et al. (2000), and 2-path cuts for the VRPTW by Kohl et al. (1999). Most approaches for routing problems embed the cutting-plane generation within the classical branch-and-bound framework for solving MIPs to produce a more sophisticated branch-and-cut procedure; the heuristics for separating violated valid inequalities are executed on the solution of the LP relaxation from the bounding phase of each tree node. The separated inequalities are then introduced into the problem formulation to strengthen the LP bound of the procedure. The proposed branch-and-cut algorithms typically begin with an edge-flow formulation and then introduce numerous existing and/or new families of valid inequalities that are tailored specifically for the type of routing problem being solved. Examples of these branch-and-cut algorithms include Padberg and Rinaldi (1991) for the TSP, Fischetti and Toth (1997) for the Asymmetric Traveling Salesman Problem (ATSP), Ruland and Rodin (1997) for the Pickup and Delivery Problem (PDP), Ascheuer et al. (2001) for the ATSPTW, Naddef and Rinaldi (2001) for the VRP, Bard et al. (2002) and Kallehauge et al. (2007) for the VRPTW, and Cordeau (2006) for the DARP.

More recently, an increased awareness for the sustainability of passenger transportation systems combined with the availability of large-scale, real-world trip datasets has shifted the focus towards optimization of car-pooling and ride-sharing services to reduce traffic congestion and pollution. Baldacci et al. (2004) studied the Car-Pooling Problem (CPP) which seeks to minimize the number of private cars used for commuting to a common workplace. They considered a variant of the problem which optimizes car pooling for the trips to the workplace independently from those for the return trips and assumes that the set of drivers and passengers are known beforehand, making it a specialization of the DARP. The effectiveness of their proposed Lagrangian column-generation method was demonstrated on instances derived from real-world data provided by a research institution in Italy. Agatz et al. (2011) contrasts the CPP with the dynamic ride-sharing problem, whereby the latter matches drivers and riders for single, non-recurring trips in real time. They proposed an optimization method which casts the problem as a graph-matching problem which is solved at regular intervals within a rolling-horizon framework. They also presented a case study which applies the approach on real-world travel-demand data from metro Atlanta. Santi et al. (2014) introduced the notion of shareability graphs as a tool to quantify the potential benefits of ride sharing, and applied it on trip data from the New York City (NYC) Taxi and Limousine Commission (TLC) trip record (NYC Taxi & Limousine Commission 2020) which stores information of more than one billion taxi rides in NYC recorded since January 2009. Alonso-Mora et al. (2017) then built on this idea to propose an anytime optimal algorithm for the on-demand ride-sharing problem, and the efficacy of their method was also demonstrated through its application on the trips from the NYC taxi dataset. Agatz et al. (2012) discusses the different planning considerations for and the issues arising from dynamic ride sharing by classifying the different variations of the problem and reviewing the optimization approaches proposed for them. Mourad et al. (2019) takes a broader view of shared mobility in their survey, whereby applications which combine transportation of people and freight in both pre-arranged and real-time settings are reviewed together with their corresponding optimization approaches.

The advent of self-driving technology combined with the race to achieve full driving automation has also spurred a growing interest in shared autonomous vehicle (SAV) systems. Advances in shared mobility services (SMS) and AV technology are widely considered to be mutually beneficial, as the widespread adoption of AVs in SMS could help make AVs financially viable (Gurumurthy and Kockelman 2018, Stocker and Shaheen 2019) and accelerate the proliferation of SMS (Thomas and Deepti 2018) at the same time. The potential impact of SAV services, ranging from their effect on the economy and the environment to the changes in policy necessary for their governance, have also been widely discussed and

reviewed in works like Milakis et al. (2017), Soteropoulos et al. (2019), and Narayanan et al. (2020). Narayanan et al. (2020) also proposed classifying SAV services as either on-demand or reservation-based systems, according to the time frame within which the trip requests are made. The former allows requests to be made in real time, making it better suited for serving dynamic trips, whereas the latter requires requests to be made in advance, making it better for recurring trips. Each has its own set of advantages: While on-demand systems can address dynamically changing trip demand, reservation-based systems can further reduce the fleet size and increase the efficiency of their routes (by reducing empty cruising time and increasing the number of customers served per vehicle), as demonstrated by Wang et al. (2014), as they know the requests ahead of time and can optimize trips over a longer time horizon. Several optimization approaches have been proposed for both systems. For on-demand systems, Farhan and Chen (2018) proposed a three-step approach for optimizing a fleet of SAVs that serves on-demand trips. It first discretizes the time horizon into 5-minute intervals, clusters trip requests from each interval by assigning riders to the nearest vehicles, and finds the optimal vehicle routes by modeling the requests from each cluster as a VRPTW and solving the problem using a tabu-search metaheuristic. On the other hand, Pinto et al. (2020) considered integrating SAVs with an existing public transit system to better serve lower density areas by using a bi-level modeling framework to jointly optimize the transit network schedule together with the sizing of the AV fleet. They proposed an iterative heuristic which solves a transit-network frequency-setting problem using a non-linear solver in the upper level and solves a dynamic combined mode choice-traveler assignment problem using an agent-based simulation in the lower level. For reservation-based systems (which are similar to the system proposed in this study), Ma et al. (2017) proposed an approach to optimize a fleet of SAVs for trip requests that are known ahead of time. However, their approach only allows vehicle sharing whereby each trip is served without being interrupted by other trip requests. This restriction admits an LP model for the problem which can then be solved efficiently. The modeling technique, however, is not applicable to ride-sharing problems like the one considered in this study. Bongiovanni et al. (2019) considers a variant of the DARP that uses electric AVs called the e-ADARP. It extends the classical DARP by incorporating additional considerations, like battery management and intermediate stops for vehicle recharging, that only apply to the operation of electric AVs. They proposed two- and three-index formulations for the problem which are then solved using a traditional branch-and-cut approach which incorporates new, problem-specific valid inequalities. They demonstrated the approach's ability to produce optimal solutions for instances with up to 40 trip requests. This study, however, considers instances that are five times larger and therefore requires a more robust approach. Numerous other

works have touted the potential benefits of these SAV systems, ranging from reducing traffic (Martinez and Viegas 2017, Alazzawi et al. 2018, Salazar et al. 2018), to increasing road capacity (Friedrich 2015, Tientrakool et al. 2011, Talebpour and Mahmassani 2016, Mena-Oreja et al. 2018, Olia et al. 2018), to reducing parking demand (Zhang et al. 2015, Dia and Javanshour 2017, Zhang and Guhathakurta 2017). However, there also appears to be a consensus that the benefits require AV adoption reaching a critical mass before they can be truly realized.

## 1.4 Dissertation Overview

Each chapter in this dissertation is intended to provide a self-contained description of the work from a specific publication. Chapters 2, 3, and 4 consider numerous aspects of solving the ride-sharing problem for commute trips with *conventional vehicles*; they focus on the optimization of routing plans whereby the trips are served by vehicles that are driven by the commuters themselves. On the other hand, Chapters 5 and 6 approach the same ride-sharing problem from a different angle; they assume the availability of *autonomous vehicles* to serve the commute trips and concentrate on how fleets of SAVs can be capitalized to generate optimal routing plans. The computational evaluations of the algorithms described in every chapter, however, center around problem instances that are derived from the Ann Arbor commute-trip dataset. The results, therefore, do not only demonstrate the efficiency and performance of the various algorithms, but they also allow the extraction of numerous metrics from a real case study that are consequential to the operation of a car-pooling or ride-sharing platform, e.g., measurements of potential reductions to the number of vehicles used, of savings in vehicle miles traveled, and of increases to average commute times.

Chapter 2 describes a community-based trip sharing approach. Based on the work in Hasan et al. (2018), the approach explores several optimization models that attempt to leverage the structure of commuting patterns and urban communities to maximize ride- and car-sharing. Several models that each enforce different driver- and passenger-matching requirements are proposed: Each implements a different set of guiding principles that describe factors that are consequential to individual commuting decisions. The common theme across all models is that the cars are driven by the commuters themselves, and the goal is to evaluate the "price" of implementing the various guiding principles in terms of vehicle reduction capability. The chapter also introduces a spatial clustering algorithm into the optimization workflow which, in addition to ensuring that the riders commute with those living in close proximity to them, provides a convenient decomposition mechanism that facilitates computational tractability. The work also describes an efficient route-

enumeration algorithm that exhaustively searches for all feasible routes that are required by the optimization models. The computational evaluations from the chapter reveal a key insight: that flexibility in driving and sharing preferences is critical to maximizing trip aggregation. The model that requires the riders to adopt different roles and to ride with different drivers and passengers daily produced the best vehicle reduction potential, which indicates that the riders will have to relax some of their commuting preferences—and thus incur some psychological discomfort—in order to increase the shareability of their trips.

Chapter 3 provides a more in-depth investigation into the car-pooling model from Chapter 2 that produced the best vehicle reduction performance. Based on the work in Hasan et al. (2020), the chapter refines the model, proposing a lexicographic objective function that first minimizes the number of vehicle utilized and then minimizes their total travel distance, and formalizes the model as the Commute Trip-Sharing Problem (CTSP). It also carefully examines the constraints that are imposed by the problem, contrasting them against those of the more popular DARP, and suggests an alternate solution method: a branch-and-price algorithm. Driven by a desire for an approach that better scales to larger vehicle capacities or to problems with more riders, the proposed method uses column generation to search for feasible routes on demand as opposed to exhaustive enumeration. It incorporates a novel wait-time relaxation technique in its pricing subproblem and a bi-level branching strategy that first branches on driver selection and then on the order of serving riders. Finally, the chapter also explores an alternate method for decomposing large-scale problems: a spatio-temporal clustering algorithm that groups commuters based on the *spatial and temporal proximity* of their trips. The computational experiments from the chapter unearthed some new insights. Not only did the results demonstrate a potential to reduce daily vehicle counts by up to 57%, but further analysis of the optimal routes revealed a critical shortcoming: The routes' inherently short nature, induced by the spatial and temporal constraints on the drivers' trips, limit the possibility for even more trip aggregation. The results also showed that a column-generation heuristic, obtained by only solving the root node of the branch-and-price tree, is capable of producing high-quality solutions within a short time frame.

Chapter 4 considers a realistic, operational setting for the CTSP by introducing some uncertainty into the commuter trip schedules. It addresses the following question: How should the model respond to changes in the return trip schedules, which occur later in the day, when the drivers of the trips need to be committed earlier in the day before the schedule changes occur? In other words, the goal is to find a robust driver assignment, one that can cover as many return trips as possible, despite the uncertainty in their schedules. The chapter describes the work by Hasan and Van Hentenryck (2020) which leverages a

scenario-sampling technique: It assumes the availability of historical trip-schedule data for each commuter to which a probability distribution can be fit and then sampled from. The sampling of every commuter distribution produces a sampled scenario. The technique is then incorporated into a two-stage optimization approach. The first solves a model that is defined on a set of sampled return-trip scenarios to assign a day's drivers and optimize their routes to the workplace. The second stage re-optimizes the return routes once the return-trip schedules have been confirmed for a setting where they are finalized by a deadline, or it re-optimizes them within a rolling-horizon approach for a setting where the schedules are confirmed in real time. Computational simulations using real trip data confirm the effectiveness of the approach: They reveal an increase in robustness of the routing plans as more sampled scenarios are utilized in the first stage. There was a price to the improved robustness however; it came with a corresponding increase in the required vehicle count. A method to formally evaluate the trade-off between robustness and vehicle reduction was then proposed.

Chapter 5 focuses on tackling the main limitation of the CTSP: its short routes which restrict its ride-sharing potential, i.e., its ability to further reduce the daily vehicle counts. Based on the work by Hasan and Van Hentenryck (2021a), the chapter considers utilizing AVs to circumvent all the driver-related constraints that were limiting the CTSP's trip aggregation potential. It formalizes the problem of serving the commute trips with AVs as the Commute Trip-Sharing Problem for Autonomous Vehicles (CTSPAV): The problem still utilizes the same lexicographic objective function of the CTSP, and its routes are also still subjected to the same spatial and temporal constraints of the riders' trips. Elimination of the driver-related constraints introduces a new set of complexities: As the AVs can serve trips throughout the day, their routes are expected to be significantly longer and thus algorithmically harder to discover. The work therefore investigates two solution approaches to the problem. The first considers a model that exploits the spatial structure of the commute-trip dataset. It solves a scheduling problem that chains together mini routes: shorter, single-direction routes with distinct pickup, transit, and drop-off phases. A CTSPAV column-generation procedure is then proposed to find a high-quality solution for the model. The second approach reduces the CTSPAV into a DARP; it then uses a more traditional DARP column-generation procedure to solve the problem. Computational evaluations revealed that each approach has complementary performance trade-offs. The first finds strong integer solutions quickly but generates weak lower bounds. On the other hand, the second approach is much slower and therefore cannot find good integer solutions within a reasonable time frame; however, it also consistently produces stronger primal and dual lower bounds. The results also confirmed the initial expectations of the AV route lengths.

For instance, within city limits, the CTSPAV routes covered an order of magnitude more trips than the CTSP's. The longer routes consequently permitted the utilization of significantly fewer vehicles to cover the same number of trips: They allowed a staggering 92% reduction in the daily vehicle counts, improving upon the results of the CTSP by 35%.

Chapter 6 takes the work on the CTSPAV a step further by attempting to synthesize a method that can leverage the unique strengths of the two methods considered in Chapter 5, i.e., one that can quickly find strong integer solutions and produce strong lower bounds at the same time. Based on the work by Hasan and Van Hentenryck (2021b), it investigates a novel, dual-modeling approach to find exact solutions to the CTSPAV. On one hand, the approach solves a CTSPAV MIP that exploits the spatial structure of the dataset by chaining together exhaustively enumerated mini routes to find strong integer solutions. On another, it solves the LP relaxation of a DARP formulation that minimizes the vehicle count to produce strong primal and dual lower bounds to the problem's primary objective. The approach solves the CTSPAV MIP using a branch-and-cut procedure and the DARP LP using column generation in parallel, all the while asynchronously transmitting new lower bounds from the DARP LP to the CTSPAV MIP to be used in valid inequalities for the branch-and-cut procedure. This approach is then compared against a more conventional branch-and-cut procedure that uses other well-established families of valid inequalities that are tailored for routing problems as well as against the CTSPAV column-generation heuristic from Chapter 5. The computational results highlight the effectiveness of the dual-modeling approach: Not only was it able to successfully close the optimality gap for several large- and medium-sized problem instances as well as those for all tight instances considered, it also consistently outperformed the other two methods it was pitted against. Furthermore, a deeper analysis of the results revealed several new insights on the benefits and drawbacks of the CTSPAV platform. In addition to significantly reducing the vehicle count, it can also reduce traffic congestion by more effectively aggregating multiple rides per vehicle. The utilization of AVs, however, will also necessarily introduce some empty miles (vehicle miles traveled without any passengers onboard) into the total travel distance.

Finally, Chapter 7 concludes the dissertation by providing a summary of the findings from each chapter and suggesting some future research directions.

# CHAPTER 2

# Community-Based Trip Sharing for Urban Commuting

## 2.1 Introduction

Car-pooling services provide an appealing alternative for urban mobility due to their potential benefits, be it in reducing traffic congestion, energy consumption, greenhouse gas emissions, or parking utilization. For instance, a case study on the CarLink car-pooling program of about 50 people revealed up to 43.5% reduction in the number of single occupant vehicle trips, a 23 miles reduction in average commute vehicle travel distance per day, and reduced parking utilization (Shaheen and Rodier 2005). Private cars however have remained as the primary choice for daily commuting due to a number of challenges associated with car pooling. For instance, a survey by Li et al. (2007) indicated difficulty in finding people with matching schedules and locations as the primary reason for not car pooling. This highlights the potential for matching platforms which alleviate this burden and automatically identify commuting groups based on factors determined to be consequential to individuals' commuting decisions. A meta-analysis of related work reveals the following set of guiding *principles* that should ideally be supported by car-pooling and car-sharing platforms:

1. Spatial proximity of riders (Richardson and Young 1981, Buliung et al. 2009).
2. Temporal proximity of riders (Tsao and Lin 1999, Buliung et al. 2010, Poulenez-Donovan and Ulberg 1994).
3. Guaranteed ride back home (Correia and Viegas 2011).
4. Low coordination costs (Arning et al. 2013).
5. Low trust concerns (Arning et al. 2013, Correia and Viegas 2011).
6. Clear commuter roles (Buliung et al. 2010, Richardson and Young 1981).

The first two principles reduce per-trip costs by matching commuters based on their schedules and locations. The third principle highlights the importance of accounting for

the commute needs for the *entire* day–individuals who cannot be matched for the return trip should not be matched for the incoming one. Principles (4–6) account for various psychological factors by limiting the perceived coordination costs, by alleviating trust concerns, and by assigning clear commuter roles to individuals.

To address these challenges, this chapter explores the concept of *community-based trip sharing* which uses the structure of communities and commuting patterns to optimize trip sharing for urban communities. Community-based trip sharing identifies matches according to the schedules and locations of riders and guarantees a ride home, hence satisfying guiding Principles (1–3) by construction. The implementation of community-based trip sharing first clusters commuters by communities before applying an optimization model to determine optimal routing plans minimizing daily car usage. Community-based trip sharing can be applied both to car pooling, where commuters use their own cars, and to car sharing, where a community has at its disposal a pool of cars for commuting purposes.

This chapter also studies the cost of implementing Principles (4–6). The implementation of each of these principles reduces the opportunities for trip sharing and the trade-off between the effectiveness of a trip-sharing platform and these guiding principles is largely unexplored. To provide new insights on this issue, this study proposes a series of optimization models for community-based trip sharing that incrementally enforce additional constraints to implement these principles. For instance, Principle (6) forces a given commuter to be either a driver or a passenger in all her trips, which may minimize opportunities for trip sharing as her schedule may vary on different days.

This study evaluates the potential and limitations of community-based trip sharing on a large case study using the Ann Arbor commute-trip dataset which contains trip data from 15,000 commuters traveling to the downtown area of the city of Ann Arbor, Michigan, over the span of a month. Ann Arbor is facing significant pressure on its downtown parking lots and congestion has been increasing annually. The results indicate that community-based trip sharing may reduce daily car usage by as much as 61% while implementing Principles (1–3). However, the benefits continuously decrease as Principles (4–6) are implemented, up to a point where they become almost negligible. This highlights the trade-off between the effectiveness of trip sharing and the (psychological) comfort of commuters.

The main contributions of this study are as follows:

1. Community-based trip sharing is introduced and applied to both car pooling and car sharing.
2. An effective implementation of community-based trip sharing is proposed, which combines clustering and optimization to minimize daily car usage.
3. Community-based trip sharing is evaluated with a large-scale, high-fidelity case study

of car pooling and car sharing for commuting purposes.

4. The study provides compelling quantitative evidence for the inherent trade-off between the benefits of trip sharing and the psychological burden imposed on commuters.

The rest of this chapter is organized as follows. Section 2.2 formally specifies the community-based trip sharing algorithm while Sections 2.3 and 2.4 present the optimization models for car pooling and car sharing respectively. Section 2.5 presents the experimental setup and results. Finally, Section 2.6 provides concluding remarks and future research directions.

## 2.2 Community-Based Trip Sharing

The community-based trip sharing algorithms aim to produce optimal routing plans that minimize the number of cars needed daily to cover all the commute trips of a set of commuters $\mathcal{C}$ for a set of days $\mathcal{D}$ subject to specific driver- and commuter-matching constraints. It uses the daily commute trip requests of $\mathcal{C}$ as input and proceeds in three major stages: (1) It clusters the commuters based on the spatial proximity of their home locations, (2) it identifies all feasible routes for each cluster, and (3) it solves an optimization model to obtain the optimal routing plan for each cluster. This section focuses on steps (1–2). The next two sections present the optimization models.

### 2.2.1 Clustering

Community-based trip sharing first clusters commuters residing in close proximity to each other to produce artificial neighborhoods within which trip sharing is considered exclusively, implementing Principle (1) from the introduction. Trip sharing is only considered intra-cluster to foster intra-community interactions and limit the distance traveled by the drivers when picking up or dropping off passengers. While it must be acknowledged that this approach precludes the acquisition of a global optimal solution, it is a necessary trade-off in this study as initial evaluations have revealed that a global solution cannot be obtained for the dataset considered within a time frame that is reasonable for an operational setting. The clustering approach not only improves tractability by decomposing the problem into smaller, independent subproblems, it also allows trip sharing for each cluster to be optimized concurrently. This strategy is also in line with the conclusion by Agatz et al. (2012) which recognizes the necessity of effective decomposition techniques to make large-scale problems computationally feasible.

This clustering algorithm aims to produce clusters of approximately equal size by

grouping no more than $N$ commuters into each cluster based on the spatial proximity of their home locations. It treats commuters as points in $\mathbb{R}^2$ whose positions are specified by the Cartesian coordinates of their homes. The algorithm itself is similar to the $k$-means clustering algorithm (Lloyd 1982), with a small exception to its assignment step. For any $N$, the number of clusters, $k = \lceil |\mathcal{C}|/N \rceil$, is first calculated. The centers for each cluster are then initialized using the $k$-means++ method by Arthur and Vassilvitskii (2007). A center $\boldsymbol{u}_1$ is first selected uniformly at random from $\mathcal{C}$. Let $S(\boldsymbol{x})$ denote the Euclidean distance between point $\boldsymbol{x}$ to the nearest center already selected. The $i^{\text{th}}$ center $\boldsymbol{u}_i$ is then selected from $\mathcal{C}$ with probability $S(\boldsymbol{u}_i)^2/(\sum_{\boldsymbol{c}\in\mathcal{C}} S(\boldsymbol{c})^2)$ until $k$ centers are obtained.

Once the cluster centers are initialized, the points are assigned to their nearest cluster centers. This assignment step is similar to that of the $k$-means clustering algorithm, except that it does so subject to a constraint that each cluster is assigned at most $N$ points. Let $\mathcal{U}$ denote the set of all cluster centers and $S(\boldsymbol{x}, \boldsymbol{y})$ the Euclidean distance between points $\boldsymbol{x}$ and $\boldsymbol{y}$. The assignment step is performed by solving the following generalized-assignment problem:

$$\min \quad \sum_{\boldsymbol{c}\in\mathcal{C}} \sum_{\boldsymbol{u}\in\mathcal{U}} S(\boldsymbol{c}, \boldsymbol{u}) X_{\boldsymbol{c},\boldsymbol{u}} \tag{2.1}$$

$$\text{s.t.} \quad \sum_{\boldsymbol{u}\in\mathcal{U}} X_{\boldsymbol{c},\boldsymbol{u}} = 1 \qquad \forall \boldsymbol{c} \in \mathcal{C} \tag{2.2}$$

$$\sum_{\boldsymbol{c}\in\mathcal{C}} X_{\boldsymbol{c},\boldsymbol{u}} \leq N \qquad \forall \boldsymbol{u} \in \mathcal{U} \tag{2.3}$$

$$X_{\boldsymbol{c},\boldsymbol{u}} \in \{0, 1\} \qquad \forall \boldsymbol{c} \in \mathcal{C}, \forall \boldsymbol{u} \in \mathcal{U} \tag{2.4}$$

The model is defined in terms of a binary variable $X_{\boldsymbol{c},\boldsymbol{u}}$ which indicates whether commuter $\boldsymbol{c}$ is assigned to cluster center $\boldsymbol{u}$. Objective function (2.1) minimizes the total distance between the commuters and their assigned cluster centers. Constraints (2.2) ensure each commuter is assigned to one cluster center, while constraints (2.3) limit the number of commuters assigned to each cluster center by $N$.

After the assignment step, the coordinates of each cluster center are updated with the mean of the coordinates of all assigned commuters:

$$\boldsymbol{u} = \frac{\sum_{\boldsymbol{c}\in\mathcal{C}} X_{\boldsymbol{c},\boldsymbol{u}} \boldsymbol{c}}{\sum_{\boldsymbol{c}\in\mathcal{C}} X_{\boldsymbol{c},\boldsymbol{u}}} \qquad \forall \boldsymbol{u} \in \mathcal{U} \tag{2.5}$$

The assignment and update steps are repeated until the assignments stabilize, i.e., until the commuter-cluster center assignments stop changing, at which point the algorithm is terminated.

### 2.2.2 Trip Sharing

This section describes the Route-Enumeration Algorithm (REA) which is used to enumerate the sets of all feasible inbound, outbound, and round-trip routes for day $\gamma$, denoted by $\Omega_\gamma^+$, $\Omega_\gamma^-$, and $\Omega_\gamma^{rt}$ respectively. The REA is applied on each day $\gamma \in \mathcal{D}$ and on each cluster before the enumerated sets are used by the optimization models in the very last stage. The REA is inspired by Santi et al. (2014) which introduced the notion of shareability networks as a tool to quantify the shareability potential of a set of trips, $\mathcal{T}$, for a fleet of shared vehicles with capacity $K$. The shareability network is a $K$-bounded hypergraph $\mathcal{H} = (\mathcal{T}, \mathcal{E})$. The nodeset $\mathcal{T}$ represents a set of trips while the set of hyperedges $\mathcal{E}$ contains all non-empty subsets of *shareable trips* from $\mathcal{T}$ with a maximum cardinality of $K$. A set of trips $\mathcal{T}_i \subset \mathcal{T}$ is shareable and can therefore be combined to form a hyperedge in $\mathcal{E}$ if and only if $|\,\mathcal{T}_i\,| \leq K$ and there exists a *feasible route* that covers all the trips in $\mathcal{T}_i$.

The REA formalizes a procedure for enumerating the feasible routes for each hyperedge in $\mathcal{E}$ of such a hypergraph whose nodeset $\mathcal{T}$ contains all the commute trips from a day and a cluster. Without loss of generality, the following describes the REA for $\Omega_\gamma^+$ for a single cluster and day $\gamma$. For the rest of this chapter, let $\mathcal{C}$ be the set of commuters from a cluster, $\mathcal{T}_\gamma^+$, $\mathcal{T}_\gamma^-$, and $\mathcal{T}_\gamma^{rt}$ denote the set of all inbound, outbound, and round trips taken by $\mathcal{C}$ on day $\gamma$ respectively, i.e., $\mathcal{T}_\gamma^+ = \{t_{c,\gamma}^+ : c \in \mathcal{C}\}$, $\mathcal{T}_\gamma^- = \{t_{c,\gamma}^- : c \in \mathcal{C}\}$, and $\mathcal{T}_\gamma^{rt} = \{t_{c\gamma}^{rt} : c \in \mathcal{C}\}$, and $\varsigma_{(i,j)}$ denote the *distance* of the shortest path between locations $i$ and $j$.

Algorithm 1 summarizes how $\Omega_\gamma^+$ is obtained from $\mathcal{T}_\gamma^+$ and vehicle capacity $K$. Routes of all individual trips from $\mathcal{T}_\gamma^+$ are first added to $\Omega_\gamma^+$ (lines 2–3). To obtain feasible routes covering more than 1 trip, an index $k$ is first set to the desired number of shared trips after which all $k$-combinations of trips from $\mathcal{T}_\gamma^+$ (denoted by $\mathcal{Q}_k$) are enumerated (lines 4–5). For each trip combination $q \in \mathcal{Q}_k$, the set of valid routes for the combination, $\Omega_q^v$, is then enumerated. For instance, for $k = 2$, $q = \{t_1, t_2\}$, $t_1 = \{o_1, dt_1, d_1, at_1\}$, and $t_2 = \{o_2, dt_2, d_2, at_2\}$, $\Omega_q^v = \{\, o_1 \rightarrow o_2 \rightarrow d_2 \rightarrow d_1,\, o_2 \rightarrow o_1 \rightarrow d_1 \rightarrow d_2 \,\}$. Let $\mathcal{C}_q$ denote the set of all riders making the trips in $q$. The algorithm then iterates over every rider $c \in \mathcal{C}_q$ and considers only routes in $\Omega_q^v$ where $c$ is the driver, i.e., $\{r \in \Omega_q^v : D_r = c\}$ (lines 8–10). The feasibility of every route in the set is then checked using the $feasible(r)$ function, and routes that are feasible are stored in a temporary set $\Omega_{\text{temp}}$. Only the route with the shortest total distance from $\Omega_{\text{temp}}$ is then added to $\Omega_\gamma^+$ (line 13). Note that this step is optional. It is done to reduce the size of $\Omega_\gamma^+$ using the fact that only a single route for each driver covering $\mathcal{C}_q$ can be used in a routing plan, so the one with the minimal total distance is selected by the REA. The procedure of exploring all $k$-combinations is repeated with incremental values of $k$ from 2 up to $K$ to completely enumerate $\Omega_\gamma^+$.

At a high level, the REA checks the shareability of every non-empty subset of trips

**Algorithm 1** REA for $\Omega_\gamma^+$

**Require:** $\mathcal{T}_\gamma^+, K$

1: $\Omega_\gamma^+ \leftarrow \emptyset$
2: **for** each $t_{c,\gamma}^+ \in \mathcal{T}_\gamma^+$ **do**
3: $\quad \Omega_\gamma^+ \leftarrow \Omega_\gamma^+ \cup \{o_{c,\gamma}^+ \rightarrow d_{c,\gamma}^+\}$
4: **for** $k = 2$ to $K$ **do**
5: $\quad \mathcal{Q}_k \leftarrow \{\text{all } k\text{-combinations of } \mathcal{T}_\gamma^+\}$
6: $\quad$ **for** each $q \in \mathcal{Q}_k$ **do**
7: $\quad\quad \Omega_q^v \leftarrow \{\text{all valid routes of } q\}$
8: $\quad\quad$ **for** each $c \in \mathcal{C}_q$ **do**
9: $\quad\quad\quad \Omega_{\text{temp}} \leftarrow \emptyset$
10: $\quad\quad\quad$ **for** each $r \in \Omega_q^v : D_r = c$ **do**
11: $\quad\quad\quad\quad$ **if** $feasible(r)$ **then**
12: $\quad\quad\quad\quad\quad \Omega_{\text{temp}} \leftarrow \Omega_{\text{temp}} \cup \{r\}$
13: $\quad\quad\quad \Omega_\gamma^+ \leftarrow \Omega_\gamma^+ \cup \{\arg\min_{r \in \Omega_{\text{temp}}} \sum_{(i,j) \in r} \varsigma_{(i,j)}\}$
14: **return** $\Omega_\gamma^+$

from $\mathcal{T}_\gamma^+$ with a maximum cardinality of $K$. It does so by considering every $k$-combination of trips up to $k \leq K$. The algorithm therefore iterates over $\mathcal{O}(n^K)$ trip combinations, where $n = |\mathcal{C}|$ represents the size of a cluster, as for a fixed vehicle capacity $K$, $\binom{n}{K} = \mathcal{O}(n^K)$. For each iteration, in which it considers a combination of trips $q$, the algorithm considers every rider $c \in \mathcal{C}_q$ as a potential driver (therefore, up to $K$ different drivers per combination) and then searches for the shortest feasible route for each driver. The search for the shortest feasible route considers up to $(2(K-1))!/(2^{K-1})$ possible location permutations per driver; for a $K$-combination of trips, a designated driver visits another $2(K-1)$ locations, and the precedence constraint requiring origins to be visited before destinations reduces the $(2(K-1))!$ location permutations by a factor of $1/(2^{K-1})$ (a complete proof to this is provided by Ruland and Rodin (1997)). Finally, the algorithm checks the feasibility of each route permutation using the $feasible(r)$ function. There are several ways of implementing the function, e.g., Gschwind and Irnich (2015) proposed a feasibility test with an $\mathcal{O}(K^2)$ time complexity. Therefore, *for each iteration*, the REA that uses such a feasibility test will have a time complexity of $\mathcal{O}(K^3(2(K-1))!/(2^{K-1}))$. Furthermore, as only the shortest feasible route for each potential driver is added to $\Omega_\gamma^+$, *each iteration* produces $\mathcal{O}(K)$ routes. Nevertheless, as both the time complexity and the number of routes generated per iteration are functions of $K$ which is fixed in this study, the total time and space complexity of the entire REA reduces to $\mathcal{O}(n^K)$.

In practice, the search procedure on lines 7–13 may be implemented efficiently using a depth-first search procedure that uses the length of shortest feasible route discovered to

Table 2.1: Summary of Optimization Models for Trip Sharing.

| Application | Name | Constraints | Principles |
|---|---|---|---|
| Ride sharing | MIP-DD | • Drivers of inbound and outbound routes identical for any given day | (1–3) |
| | MIP-DD-DIO | • Commuters of inbound and outbound routes identical for any given day | (1–5) |
| | MIP-WD-DIO | • Commuters of inbound and outbound routes identical for any given day<br>• Drivers identical every day | (1–6) |
| | MIP-WD-WIO | • Commuters of inbound and outbound routes identical for any given day<br>• Drivers identical every day<br>• Passenger-driver pairings identical every day | (1–6) |
| Car sharing | MIP-DC | • Total number of inbound and outbound routes identical for any given day | (1–3) |

prune the search tree. Moreover, since the search procedure for all $q \in \mathcal{Q}_k$ are independent of each other, they may be executed concurrently in parallel. The same algorithm may be repeated on $\mathcal{T}_\gamma^-$ to enumerate $\Omega_\gamma^-$, and only a minor modification is needed for the algorithm to enumerate $\Omega_\gamma^{rt}$. On lines 9–13, the algorithm essentially searches for the shortest feasible route covering the *inbound trips* of $\mathcal{C}_q$ with $c$ as the designated driver. To enumerate $\Omega_\gamma^{rt}$, the search procedure is extended so that it then searches for the shortest feasible route covering the *outbound trips* of the same set of commuters $\mathcal{C}_q$ with the same driver $c$. The pair of feasible routes covering the *inbound* and *outbound* trips of $\mathcal{C}_q$ with $c$ as the designated driver represents a feasible *round-trip* route that can then be added to $\Omega_\gamma^{rt}$. The algorithm would therefore require $\mathcal{T}_\gamma^{rt}$ to enumerate $\Omega_\gamma^{rt}$.

## 2.3 Optimization Models for Ride Sharing

This section presents the optimization models for finding the optimal routing plan for each cluster and for every day $\gamma \in \mathcal{D}$. The models utilize the sets of feasible routes enumerated from the previous section. The names, high-level constraints, and adopted guiding principles of each model are summarized in Table 2.1.

### 2.3.1 MIP-DD

MIP-DD is the least constrained optimization model for ride sharing and satisfies Principles (1–3) from the introduction. It minimizes the number of cars required subject to the constraint that the set of drivers are identical for the inbound and outbound routes. This ensures the cars leaving a cluster returns to the cluster every day. The model optimizes the routing plan for each day independently. As a result, the set of drivers selected for different days do not need to be identical. Passengers also do not need to be matched with the same driver for inbound and outbound routes on the same day or on different days.

The model is defined in terms of binary variable $X_r$ which indicates whether route $r \in \Omega_\gamma^+ \cup \Omega_\gamma^-$ is selected for the optimal plan of day $\gamma$. The model for day $\gamma$ is specified as follows:

$$\min \quad \sum_{r \in \Omega_\gamma^+ \cup \Omega_\gamma^-} X_r \tag{2.6}$$

$$\text{s.t.} \quad \sum_{r \in \Omega_\gamma^+ : c \in \mathcal{C}_r} X_r = 1 \qquad \forall c \in \mathcal{C} \tag{2.7}$$

$$\sum_{r \in \Omega_\gamma^- : c \in \mathcal{C}_r} X_r = 1 \qquad \forall c \in \mathcal{C} \tag{2.8}$$

$$\sum_{r \in \Omega_\gamma^+ : D_r = c} X_r = \sum_{r \in \Omega_\gamma^- : D_r = c} X_r \qquad \forall c \in \mathcal{C} \tag{2.9}$$

$$X_r \in \{0, 1\} \qquad \forall r \in \Omega_\gamma^+ \cup \Omega_\gamma^- \tag{2.10}$$

Objective function (2.6) minimizes the number of cars used for inbound and outbound routes. Constraints (2.7) and (2.8) indicate that exactly one inbound and one outbound route must be selected for each commuter respectively. Constraints (2.9) ensure the set of drivers selected for the inbound routes are identical to that for the outbound routes.

### 2.3.2 MIP-DD-DIO

MIP-DD-DIO contains an additional requirement compared to MIP-DD. It requires that the commuters of any pair of inbound and outbound routes are identical on any day. This constraint reduces coordination costs and alleviates trust concerns by reducing the maximum unique matches per commuter from 2 to 1 per day. Hence the model can be considered to satisfy Principles (1–5), although it does so partially. To satisfy this constraint, the model utilizes $\Omega_\gamma^{rt}$ since a round-trip route already ensures the commuters of its inbound and outbound routes are identical. Similar to MIP-DD, the model optimizes the routing

plan for each day independently. It uses a single binary variable $X_r$ to indicate whether route $r \in \Omega_\gamma^{rt}$ is selected for the optimal plan. The model for day $\gamma$ is specified as follows:

$$\min \quad \sum_{r \in \Omega_\gamma^{rt}} X_r \tag{2.11}$$

$$\text{s.t.} \quad \sum_{r \in \Omega_\gamma^{rt} : c \in \mathcal{C}_r} X_r = 1 \qquad \forall c \in \mathcal{C} \tag{2.12}$$

$$X_r \in \{0, 1\} \qquad \forall r \in \Omega_\gamma^{rt} \tag{2.13}$$

Objective function (2.11) minimizes the number of cars used for the round trips and constraints (2.12) state that exactly one round-trip route must be selected for each commuter.

### 2.3.3 MIP-WD-DIO

MIP-WD-DIO has the same objective and constraints as MIP-DD-DIO, with an additional requirement that the drivers designated for every day $\gamma \in \mathcal{D}$ must be identical. In other words, a commuter is prohibited from being a driver on some days and a passenger on others. This model satisfies Principles (1–6), since drivers and passengers now have clearly defined roles. The model uses two binary variables: variable $X_r$ is the same as in MIP-DD-DIO and variable $Y_c$ indicates whether commuter $c \in \mathcal{C}$ is selected as the driver of a round-trip route. $P_r$ denotes the set of passengers of route $r$, i.e., $P_r = \mathcal{C}_r \setminus \{D_r\}$. The model is specified as follows:

$$\min \quad \sum_{\gamma \in \mathcal{D}} \sum_{r \in \Omega_\gamma^{rt}} X_r \tag{2.14}$$

$$\text{s.t.} \quad \sum_{r \in \Omega_\gamma^{rt} : c \in \mathcal{C}_r} X_r = 1 \qquad \forall \gamma \in \mathcal{D}, \forall c \in \mathcal{C} \tag{2.15}$$

$$Y_{D_r} \geq X_r \qquad \forall \gamma \in \mathcal{D}, \forall r \in \Omega_\gamma^{rt} \tag{2.16}$$

$$Y_c \leq 1 - X_r \qquad \forall \gamma \in \mathcal{D}, \forall r \in \Omega_\gamma^{rt}, \forall c \in P_r \tag{2.17}$$

$$X_r \in \{0, 1\} \qquad \forall \gamma \in \mathcal{D}, \forall r \in \Omega_\gamma^{rt} \tag{2.18}$$

$$Y_c \in \{0, 1\} \qquad \forall c \in \mathcal{C} \tag{2.19}$$

Objective function (2.14) globally minimizes the number of cars for every day $\gamma \in \mathcal{D}$. Constraints (2.15) ensure exactly one round-trip route is selected for each commuter every day, constraints (2.16) assign drivers of selected round-trip routes, and constraints (2.17) ensure passengers of selected routes are never assigned as drivers. The differences with MIP-DD-DIO are quite subtle when formalized: The key is to recognize that the univer-

sal quantification in constraints (2.16) and (2.17) forces a driver to drive every day and a passenger to never drive. Model MIP-DD-DIO, in contrast, is optimized independently for each day, therefore it does not require a commuter to be assigned exclusively as a driver or a passenger every day.

### 2.3.4 MIP-WD-WIO

MIP-WD-WIO adds a final requirement that passenger-driver pairings for every day $\gamma \in \mathcal{D}$ must be identical, i.e., a passenger must always commute with the same driver. This is the most desirable model as it strongly obeys all principles. Let $\Omega^{rt}$ denote the set of all feasible round-trip routes across all days, i.e. $\Omega^{rt} = \{r \in \Omega^{rt}_{\gamma} : \gamma \in \mathcal{D}\}$, and $\mathcal{W}$ denote the set of all passenger-driver pairs obtained from all feasible round-trip routes, i.e., $\mathcal{W} = \{(c, D_r) : c \in P_r, r \in \Omega^{rt}\}$. The model uses three binary variables: $X_r$ and $Y_c$ are the same as those used in MIP-WD-DIO, while $V_w$ keeps track of each passenger-driver pair $w \in \mathcal{W}$ selected in the optimal plan. Furthermore, let $\Gamma_c$ denote the set of all routes where $c$ is a passenger, i.e., $\Gamma_c = \{r \in \Omega^{rt} : c \in P_r\}$, and $\Lambda_c$ denote the set of all possible drivers for passenger $c$, i.e., $\Lambda_c = \{D_r : r \in \Gamma_c\}$. The objective function of the model is given by (2.14), subject to (2.15), (2.16), (2.17), (2.18), (2.19), and

$$V_{(c,D_r)} \geq X_r \qquad \forall \gamma \in \mathcal{D}, \forall r \in \Omega^{rt}_{\gamma}, \forall c \in P_r \tag{2.20}$$

$$V_{(c,p)} \leq 1 - X_r \qquad \forall \gamma \in \mathcal{D}, \forall r \in \Omega^{rt}_{\gamma}, \forall c \in P_r, \forall p \in \Lambda_c \setminus \{D_r\} \tag{2.21}$$

$$V_w \in \{0, 1\} \qquad \forall w \in \mathcal{W} \tag{2.22}$$

Constraints (2.20) select passenger-driver pairs according to selected round-trip routes, while constraints (2.21) prohibit selection of passenger-driver pairs other than those from selected round-trip routes.

## 2.4 Optimization Model for Car Sharing

This section studies community-based car sharing and assumes that each cluster has a pool of cars that can be used by anyone for commuting trips. Model MIP-DC minimizes daily car usage for commuting trips subject to the constraint that the number of inbound routes is equal to the number of outbound routes for any day. This constraint ensures that the number of cars shared in the cluster remains the same day after day. The model approximates the

number of daily cars and routes required for a car-sharing model.[1] Drivers for inbound and outbound routes on any day do not need to be identical, which makes the model even less restrictive than MIP-DD. The model optimizes the routing plan for each day independently. It uses a binary variable $X_r$ like in MIP-DD. Its objective function is given by (2.6) subject to (2.7), (2.8), (2.10), and

$$\sum_{r \in \Omega_\gamma^+} X_r = \sum_{r \in \Omega_\gamma^-} X_r \tag{2.23}$$

Constraint (2.23) ensures the total number of inbound and outbound routes are identical for any day $\gamma$. This model satisfies the same set of principles as MIP-DD.

## 2.5 Computational Results

**The Dataset** The problem instances of this chapter are constructed with trips from the Ann Arbor commute-trip dataset. The dataset provides trip information for 15,000 commuters within an area spanning 13,000 square miles. About 9,000 people commute to these parking lots on any given weekday. For additional insight, the commuters are partitioned into two sets; the 4,000 commuters living within city limits (the Ann Arbor region bounded by highways US-23, M-14, and I-94), and the 11,000 commuters living outside that region. Results are given for the busiest week of the month (week 2), and focus on the trips made on Monday–Thursday, which are the busiest days. As was shown in Figure 1.2, the commuting pattern of this population is remarkably predictable and consistent, which is a key property for effective car pooling (Buliung et al. 2010).

**Experimental Settings** This study assumes that any commuter $i$, when requesting a commute trip for any day, would specify the desired arrival time at the destination of the inbound trip, $at_i^+$, and the desired departure time at the origin of the outbound trip, $dt_i^-$. This assumption is similar to that made in other literatures, e.g., Jaw et al. (1986), Cordeau and Laporte (2003b), Cordeau (2006). On top of that, it also assumes that each tolerates a maximum time shift of $\pm\Delta$ to the desired times. Therefore, by treating the arrival and departure times to and from the parking structures as the desired times, time windows of $[a_{d_i^+}, b_{d_i^+}] = [at_i^+ - \Delta, at_i^+ + \Delta]$ and $[a_{o_i^-}, b_{o_i^-}] = [dt_i^- - \Delta, dt_i^- + \Delta]$ are associated with the destination of the inbound trip, $d_i^+$, and the origin of the outbound trip, $o_i^-$, of commuter $i$ respectively. Accordingly, letting $L_i^+$ and $L_i^-$ denote the ride-duration limit of

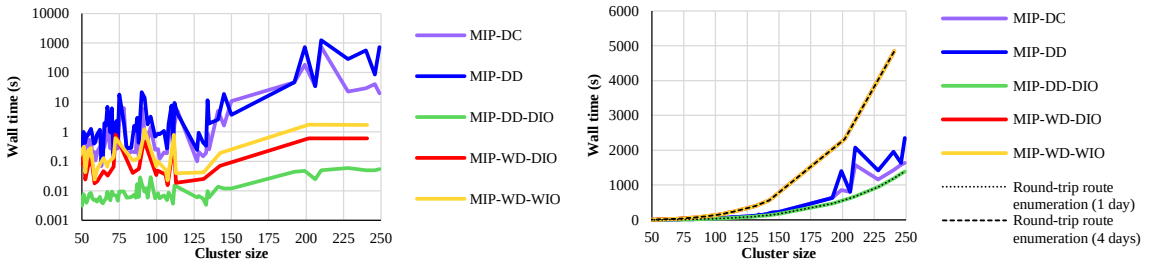
[1] For simplicity, we ignore where the cars are parked in the cluster: They can be at a central location or with the drivers. We also assume that the cars within each cluster are easily reachable by all drivers and ignore how the drivers get to the cars.

commuter $i$'s inbound and outbound trips respectively, the time windows at the origin of her inbound trip, $o_i^+$, and at the destination of her outbound trip, $d_i^-$, are defined by $[a_{o_i^+}, b_{o_i^+}] = [a_{d_i^+} - \zeta_{o_i^+} - L_i^+, b_{d_i^+} - \zeta_{o_i^+} - \tau_{(o_i^+, d_i^+)}]$ and $[a_{d_i^-}, b_{d_i^-}] = [a_{o_i^-} + \zeta_{o_i^-} + \tau_{(o_i^-, d_i^-)}, b_{o_i^-} + \zeta_{o_i^-} + L_i^-]$ respectively. Finally, similar to Hunsaker and Savelsbergh (2002), this study assumes that every commuter $i$ tolerates an $R\%$ maximum extension to the duration of her direct trip, i.e., $L_i = (1 + R)\tau_{(o_i, d_i)}$.

Unless stated otherwise, $\Delta = 10$ mins and $R = 50\%$ are used in most experiments, although sensitivity analyses to both parameters that utilize other values are also performed. Due to the non-deterministic nature of the initialization of the clustering algorithm, it is repeated 100 times for every value of $N$ considered, after which only the result with the smallest final total distance is used. Unless stated otherwise, a value of $N = 200$ is used for most experiments as it produces clusters that are sufficiently large to promote ample trip sharing, but not ones that are excessively large so as to make the problem instances intractable. However, a sensitivity analysis that utilize other $N$ values is also performed. As the study investigates the use of cars, $K$ is set to 4 in all experiments. The GPS coordinates of every address considered are geocoded using Geocodio, while the shortest path, travel time, and distance between any two locations are estimated using the GraphHopper Directions API which uses data from OpenStreetMap. All algorithms are implemented in C++ using OpenMP to handle parallelization duties, and the Gurobi 7.5.1 solver is invoked to solve all optimization models and MIPs. Every problem instance is solved on a high-performance compute cluster, utilizing 12 cores of a 2.5 GHz Intel Xeon E5-2680v3 processor, 48 GB of RAM, and a time limit of 4 hours. The optimization models for every problem instance considered could be solved to optimality within the time limit.

### 2.5.1 Computation Times

Figures 2.1(a) and 2.1(b) summarize the total computation times of each optimization algorithm for several cluster sizes when $\Delta = 10$ mins and $R = 50\%$. The clusters with different sizes are constructed by simply varying $N$ for the clustering algorithm. Figure 2.1(a) shows the wall times for solving each model exclusively, without the inclusion of the times for enumerating their routes. A logarithmic time scale is used to highlight the time differences for small cluster sizes. Figure 2.1(b) shows the *total* wall times for solving each model and for enumerating their routes on a linear time scale. For MIP-DC, MIP-DD, and MIP-DD-DIO which optimize routing plans on a daily basis, the total time shown consists of the time to solve the optimization model combined with the time to enumerate their respective sets of feasible routes ($\Omega_\gamma^+ \cup \Omega_\gamma^-$ for MIP-DC and MIP-DD, $\Omega_\gamma^{rt}$ for MIP-DD-DIO) *for trips from*




(a) Excluding Route Enumeration Times (b) Total Including Route Enumeration Times

Figure 2.1: Computation Times of Every Optimization Model.

*a single day*. Conversely, as MIP-WD-DIO and MIP-WD-WIO optimize plans for multiple days at once, their total times consist of the time to solve the optimization model *for trips from all four days considered* ($\mathcal{D} = \{\text{Monday,Tuesday,Wednesday,Thursday}\}$) combined with the time to enumerate their sets of feasible routes, $\{\Omega_\gamma^{rt} : \gamma \in \mathcal{D}\}$. And since the number of traveling commuters from each cluster is not identical every day, the cluster size for the latter two models are represented by the cluster's daily average number of travelers. For additional perspective, the time spent by the REA for enumerating $\Omega_\gamma^{rt}$ for a single day and for the four days considered are also displayed.

Figure 2.1(b) depicts the total computation time of every model increasing exponentially with cluster size. While MIP-WD-DIO and MIP-WD-WIO appear to be more expensive than the other three models in the figure, this is only because their total run times are dominated by their route-enumeration stages, which have to enumerate the feasible routes for all four days as opposed to only for a single day for the other three models. The optimization models for MIP-WD-DIO and MIP-WD-WIO alone solve almost instantaneously; their optimal solutions are consistently found in less than 2 seconds for the cluster sizes considered as shown in Figure 2.1(a). The same can also be said for MIP-DD-DIO whose total run time is completely dominated by its route-enumeration stage and whose model solves almost instantaneously. By contrast, the optimization models for MIP-DC and MIP-DD are harder to solve. In some cases, the times for solving their optimization models are comparable to those for enumerating their routes. Figure 2.1(a) shows that not only are they significantly more expensive, but their solution times also appear to increase exponentially with cluster size.

### 2.5.2 Reduction in Car Usage

Figure 2.2 summarizes car reduction results during the busiest week of the month using $\Delta = 10$ mins, $R = 50\%$, and $N = 200$ for the clustering algorithm. It shows the aggre-

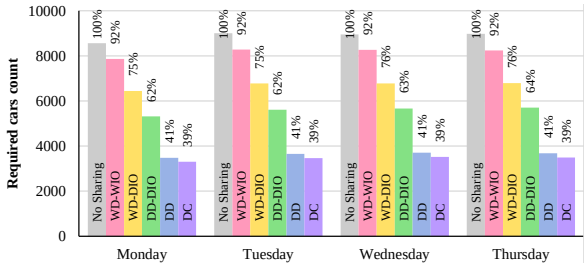


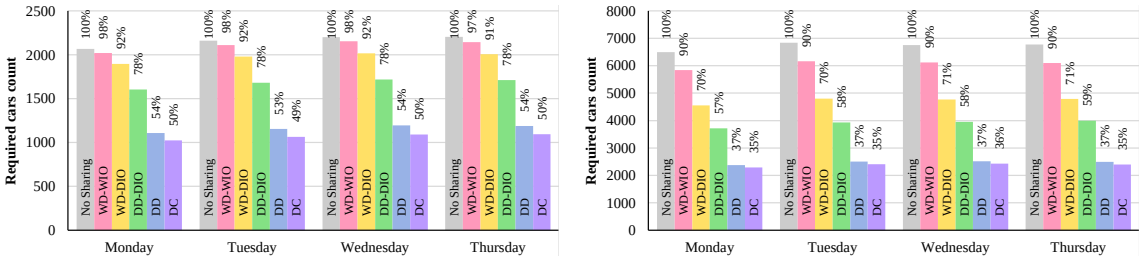

Figure 2.2: Car Reduction Results of Every Optimization Model.

gated number of required cars from all clusters for every optimization model for the first 4 weekdays of week 2. It also displays the counts as a percentage of the number of cars under the existing no-sharing conditions.

The first insight is that ride-sharing and car-sharing programs may bring substantial benefits for the city of Ann Arbor. For both programs, the results show a potential reduction of about 59% in daily car utilization for community-based ride sharing (MIP-DD) and 61% for community-based car sharing (MIP-DC). This would substantially reduce pressure on parking in the city and congestion during the morning and evening commutes.

The second insight is that these benefits require flexibility. As the models enforce additional constraints on driver selection and driver-passenger matching, the results significantly deteriorate. When the matching must be the same for inbound and outbound routes on any day (MIP-DD-DIO), the potential reduction in car utilization is around 37%. This is still significant, but these results also highlight the challenge of matching commuters in round trips versus one-way trips. When the drivers and the driver-passenger matching are identical every day (MIP-WD-WIO), the reduction falls to about 8%. It remains around 25% when the drivers are identical every day, but the driver-passenger matching must only be the same inbound and outbound each day (but may differ on different days) (MIP-WD-DIO). It is particularly interesting that desirable properties (4–6) for ride-sharing and car-sharing platforms are extremely hard to enforce while reducing car utilization effectively. Any effective platform will require a different sharing pattern for every weekday, although these schedules can be repeated week after week. As a result, these platforms will necessarily impose some psychological burden as commuters need to interact with different people and adopt different roles daily.

Figures 2.3(a) and 2.3(b) provide the car reduction results when the algorithms only consider the commuters inside or outside the city limits respectively. A quick comparison of the results inside and outside the city for each model reveals that the vehicle reduction percentage of each is larger outside the city than inside. This can be attributed to the way the temporal constraints are defined for trip-sharing feasibility in this study. While




(a) Inside City Limits

(b) Outside City Limits

Figure 2.3: Car Reduction Results of Every Model Inside/Outside City Limits.

the parameter that directly controls the size of the time windows is identical for every trip ($\Delta = 10$ mins), the ride-duration limit for every trip is defined as a constant factor of the direct-trip duration ($1.5\times$ in these results as $R = 50\%$). The absolute value of the extension to the direct-trip duration is therefore larger for longer direct trips, which is exactly the case for commuters living outside the city. The longer duration limits allow the REA to combine more trips to form more feasible routes, consequently providing every model with more routes to choose from during optimization, and finally leading to the better vehicle reduction results observed outside the city. This factor is also evidenced by the number of feasible inbound and outbound routes enumerated, $|\,\Omega_\gamma^+ \cup \Omega_\gamma^-\,|$, per cluster outside the city being approximately an order of magnitude larger on average than that inside. Otherwise, the general trend observed in Figure 2.2 is still present both inside and outside the city limits, whereby car reduction performance degrades as more driver- and commuter-matching constraints are enforced across the different models.

### 2.5.3 Reduction in Vehicle Miles Traveled

Figure 2.4 summarizes the total travel distance of the routing plans of all clusters for every optimization model. Similar to Section 2.5.2, the results are obtained using $\Delta = 10$ mins, $R = 50\%$, and $N = 200$, and the percentage of each total as a fraction of that under the existing no-sharing conditions is also displayed. The reduction in travel distance of the different models displays a trend that mimics that from Figure 2.2 for the reduction in cars utilized, whereby the least constrained model, MIP-DC, produces the most significant reduction, and the reductions decline progressively as the models implement more driver- and commuter-matching constraints. The best performing models, MIP-DC and MIP-DD, reduces the daily vehicle miles traveled by an average of 142,000 per day, which translates to approximately 16 miles per commuter. The trend observed is intuitive as the models that utilize fewer cars also produce less vehicle miles traveled due to their ability to ag-

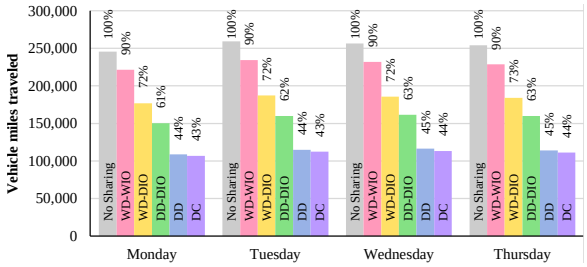
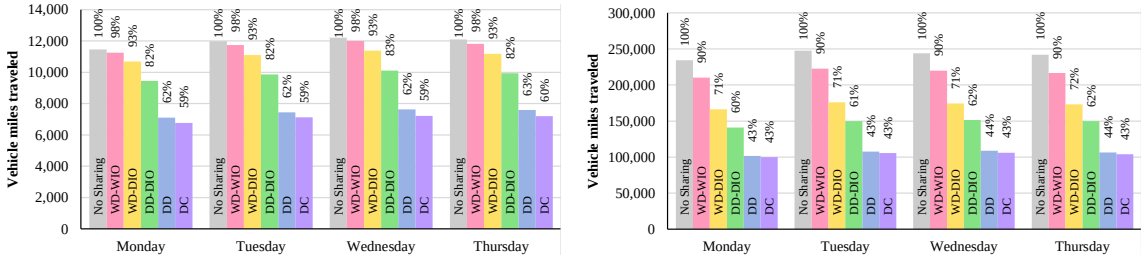



Figure 2.4: Total Travel Distance of Every Optimization Model.




(a) Inside City Limits

(b) Outside City Limits

Figure 2.5: Total Travel Distance of Every Model Inside/Outside City Limits.

gregate more trips per route. The main takeaway from the previous section therefore also applies here, where flexibility in the commuters' matching and role-adoption preferences is necessary to obtain significant reductions in vehicle miles traveled.

Figures 2.5(a) and 2.5(b) then separate the total travel distance results of clusters inside and outside city limits. The results once again mirror those for car reduction from Figures 2.3(a) and 2.3(b), whereby the reductions for every model outside city limits are greater than those inside. This can be attributed to the same reason described earlier; the larger ride-duration limits for trips originating further away from the parking structures also allow more trip aggregation which eventually improve travel distance reduction.

### 2.5.4 Car Reduction Sensitivity to $N$, $\Delta$, and $R$

Figure 2.6 summarizes the sensitivity of the car reduction results of every model to the cluster size. For this analysis, smaller clusters were first created by setting $N$ to 100 and 150 for the clustering algorithm. The routing plan for these clusters were then optimized with every model, after which their aggregated number of cars are compared against those for $N = 200$ shown earlier. The values of $\Delta$ and $R$ were kept constant at 10 mins and 50% throughout. Firstly, the results show degradations in car reductions for every model as the clusters become progressively smaller, which is not unexpected as the smaller clusters have fewer trips which consequently provide fewer intra-cluster trip-sharing opportunities. The

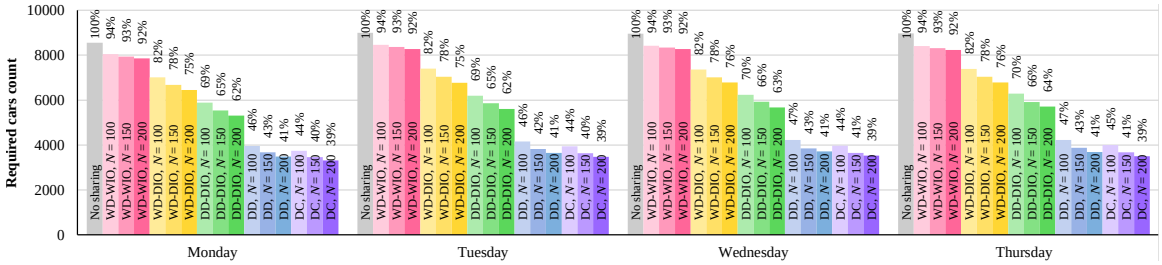



Figure 2.6: Car Reduction Results of Every Model for $N \in \{100, 150, 200\}$.

results, however, do not fundamentally change the nature of the prior conclusions. MIP-WD-WIO shows the least sensitivity to changes in $N$, whereby its results changed by only $\pm 1\%$ when $N$ is increased or decreased by 50 from 150. Both MIP-DC and MIP-DD display moderate sensitivity to $N$, whereby decreasing it by 50 from 150 degrades the results by 3–4%, while increasing $N$ by 50 improves the results by 1–2%. Finally, the intermediate models MIP-DD-DIO and MIP-WD-DIO show the most sensitivity, with degradations and improvements of about 4% and 2–3% respectively, which are not negligible, but the improvements do not bring their results close to those of MIP-DC or MIP-DD. While larger $N$ values are always desirable for increasing trip sharing, they also cause the computation times to increase exponentially as was shown in Figure 2.1(b). Therefore, its value has to be judiciously selected in an operational setting to provide ample opportunities for trip sharing while ensuring that the resulting problem instances are still tractable.

Figure 2.7 shows the car reduction performance of each model as $\Delta$ is varied between $\{5, 10, 15\}$ mins while keeping $N = 100$ and $R = 50\%$. Recall that $\Delta$ directly affects the size of the time windows of the arrival and departure times at the parking structures. Therefore, the parameter impacts the QoS of the trip-sharing platform, where smaller values allow the commuters to arrive and depart at times that are closer to their desired times. Smaller values of $\Delta$ are therefore more desirable from a QoS enhancement perspective. However, as seen in Figure 2.7, decreasing $\Delta$ by merely 5 mins from the 10 mins reference value significantly degrades the car reduction performance of most models. MIP-WD-WIO appears to be the least sensitive model again, but its results still degraded by 3–4%. MIP-DC and MIP-DD was more sensitive, having their results degrade by 7–9%, and MIP-DD-DIO and MIP-WD-DIO were the most severely affected, having double digit percentage degradations of 10–12%. Reducing $\Delta$ affected the results by making the trip schedules less flexible, consequently decreasing the number of trips that can be combined by the REA to form feasible routes and the final performance of each model. Increasing $\Delta$ by 5 mins from the 10 mins reference improved the car reduction performances of every model, however by smaller degrees compared to their degradations, signaling diminishing

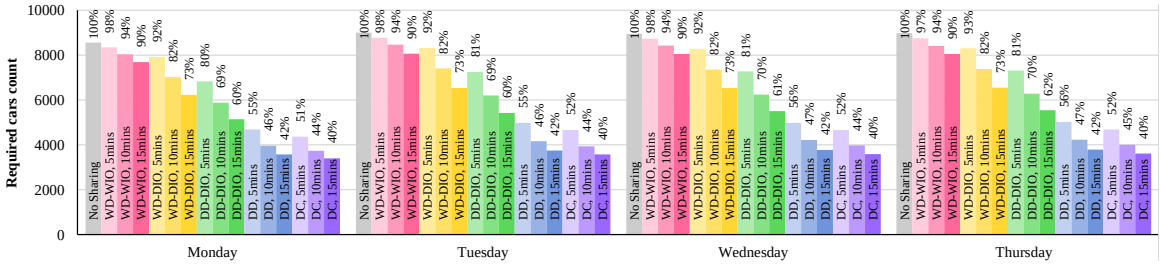



Figure 2.7: Car Reduction Results of Every Model for $\Delta \in \{5, 10, 15\}$ mins.

marginal improvements to car reduction as $\Delta$ is increased. And while increasing $\Delta$ improves trip shareability by making the trip schedules more flexible, it comes at the price of a corresponding decrease in QoS. This explains why $\Delta$ is set to 10 mins for most of the experiments, as it was deemed to still allow significant trip sharing while not severely inconveniencing the commuters.

Finally, Figure 2.8 shows the sensitivity of each model to $R$ while $N$ and $\Delta$ are kept constant at 100 and 10 mins respectively. Recall that $R$ directly influences the maximum ride duration of every commuter's trip. Thus, similar to $\Delta$, it also affects the platform's QoS and keeping $R$ as small as possible is most desirable from a QoS improvement point of view. One would also anticipate a similar trade-off between car reduction and user convenience when varying this parameter as its increase positively affects the flexibility of trip schedules, which in turn leads to performance improvements for the models as shown earlier. Unsurprisingly, such is the observation for Figure 2.8, where decreasing $R$ by 25% from the reference 50% value degrades the car reductions of every model. MIP-WD-WIO is least affected (by only 3%), MIP-WD-DIO and MIP-DD-DIO are moderately affected by 10–12%, and MIP-DD and MIP-DC are most severly affected in this case, having their performance decline significantly by 15–18%. Increasing $R$ by 25% and 50% relative to the 50% reference expectedly improve car reduction, but by smaller degrees compared to their declines, once again indicating diminishing marginal improvements to car reduction performance with increases in $R$. Once again, by increasing $R$, one is trading off user convenience for (marginal) improvements to car reduction performance. This also provides the rationale for using $R = 50\%$ in most experiments, as it was deemed to provide the best car reduction-user convenience trade-off.

### 2.5.5 Cost of Car Balancing

All models ensure that the cars leaving a cluster return to the cluster. Figure 2.9 shows that the cost of this balancing constraint is relatively small. It compares MIP-DD and MIP-

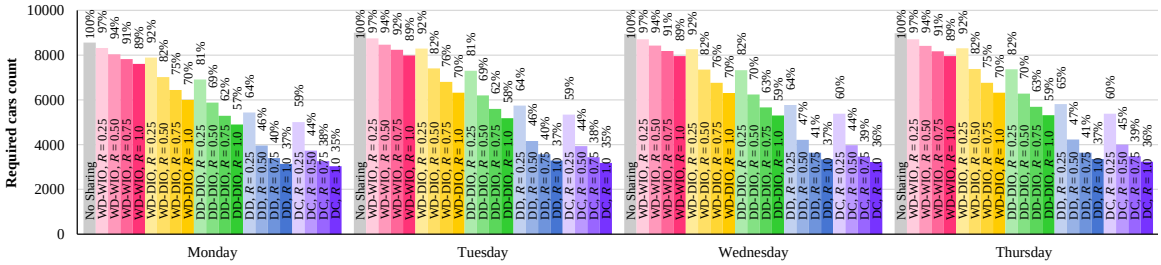



Figure 2.8: Car Reduction Results of Every Model for $R \in \{25\%, 50\%, 75\%, 100\%\}$.

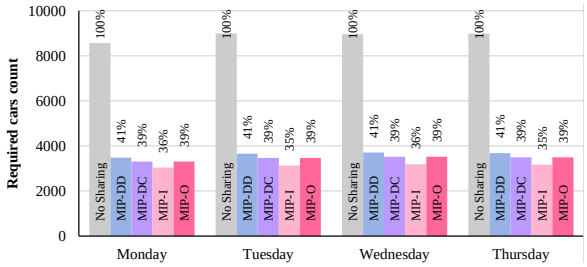



Figure 2.9: Cost of Car Balancing.

DC with two models, MIP-I and MIP-O that minimize the inbound and outbound routes independently. Balancing the cars induces a cost increase of about 2% for MIP-DD and 0% for MIP-DC over MIP-O. The balancing cost of MIP-DD is slightly higher as it not only balances the inbound and outbound car counts like MIP-DC, but it also ensures that the drivers of their inbound and outbound routes are identical. Interestingly, the results of MIP-O indicate that optimization of the outbound trips is more challenging due to their less regular trip schedules.

## 2.6 Conclusion

This chapter explored the idea of community-based trip sharing and its application to car pooling and car sharing. It studied the trade-off between the effectiveness of community-based trip sharing in reducing daily car usage and the desirable principles for trip-sharing platforms. These ideas were explored on a large case study using a dataset of 15,000 commuters working in downtown Ann Arbor, Michigan.

The results showed that a platform implementing the core principles for trip sharing can reduce daily car usage by up to 61%, which amounts to approximately 5,400 cars, and reduce traveled miles by 142,000 daily. However, as additional principles are integrated, e.g., low coordination costs and clear commuter roles, the benefits progressively decline and eventually disappear almost entirely. This study also showed that these results are ro-

bust with respect to the cluster sizes, time windows, and ride-duration extensions, although more flexibility on each dimension helps alleviate some of the trade-off, with temporal flexibility bringing the most benefits. The study thus indicates that there are trade-offs between the principles themselves.

Future work will be devoted to the maximization of trip-sharing opportunities by exploring other clustering techniques, integrating personalized matching constraints based on individual commuter preferences, and scaling the algorithms for applications in large metropolitan areas.

# CHAPTER 3

# The Commute Trip-Sharing Problem

## 3.1 Introduction

Parking occupies a significant portion of our cities. In the United States, for instance, there are at least 800 million parking spaces and, in Los Angeles County, 14% of the city space is devoted to parking (Taylor 2018). Parking also contributes to congestion: Based on a sample of 22 studies in the United States, the average share of traffic cruising to find a parking spot is 30% and the average cruising time is just under 8 minutes in downtown areas (Shoup 1997, 2006).

Parking pressure has also been steadily increasing in cities, university campuses, and corporations, alongside other concerns such as traffic congestion, fuel prices, and greenhouse gas emissions. In the city of Buffalo, New York, the overall supply of parking space has remained constant for the last 20 years, while the downtown population and the workforce have increased by 70% and 30% respectively (Epstein 2018). These parking shortages are perceived as an impediment to future economic developments, as corporations may elect to move elsewhere when growing their operations. University campuses feel similar parking pressures. For instance, Stanford University suffers from a lack of parking spaces due to construction and a growth in population (Chesley 2017). The research underlying this paper was originally motivated by parking pressure at the University of Michigan in Ann Arbor. Figure 3.1 depicts the parking utilization of the 15 most used parking lots in downtown Ann Arbor. They show a typical parking usage: Cars arrive in the morning, park in the lot for 6 to 10 hours, and leave the lot in the evening.

To address the increasing demand on these lots, we started to investigate the potential of a community-based car-pooling program in Chapter 2. The idea was to implement a car-pooling program organized around the communities commuting to the university, exploiting the knowledge of when employees were arriving in the morning and leaving in the evening. However, while car pooling has long been proposed as a solution to reduce

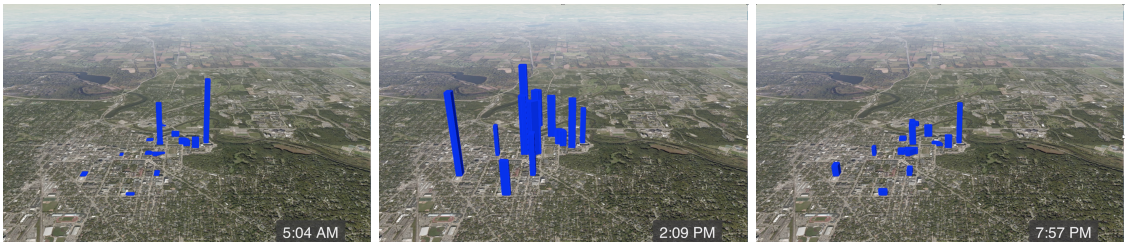



Figure 3.1: Occupancy of the Main University Parking Lots in Downtown Ann Arbor.

peak-hour congestion and parking utilization, its adoption in the US remains poor as 76.4% of American commuters chose to drive alone according to the 2013 American Community Survey (McKenzie 2015). A study on factors influencing car-pool formation by Li et al. (2007) revealed difficulty in finding people with the same location and schedule as the primary reason for not car pooling. As a result, we investigated how to alleviate this burden and studied the feasibility of a matching platform that would automatically identify commuting groups based on factors determined to be consequential to individuals' commuting decisions in Chapter 2. One of the results of our study was the recognition that an effective car-pooling platform will need to accommodate different sharing patterns for every weekday and, as a result, the platform will need to optimize trip matching on a daily basis to allow significant car pooling to occur.

The goal of this chapter is to propose and analyze scalable optimization algorithms for powering such a platform. A meta-analysis of related work reveals that car-pooling and car-sharing platforms should at least implement the following three guiding principles:

1. Spatial proximity of riders (Richardson and Young 1981, Buliung et al. 2009).
2. Temporal proximity of riders (Tsao and Lin 1999, Buliung et al. 2010, Poulenez-Donovan and Ulberg 1994).
3. Guaranteed ride back home (Correia and Viegas 2011).

The first two guidelines are natural since car pooling is unlikely to occur for riders who are not close spatially or whose schedules are not compatible. The third guideline is critical: It is unlikely that many riders will use a platform that does not guarantee a ride back home in the evening. The guarantee of a ride back home is one of the main contributions of this work: For instance, the car-pooling platform SCOOP provides only weak guarantees for "ride back" and has monthly limits on how much auxiliary services can be used when a ride back is not available. In contrast, this study approaches the matching of riders in two steps. In the first step, riders are grouped into neighborhoods using a clustering algorithm. In the second step, an optimization algorithm selects drivers and matches riders to minimize the number of cars and the total travel distance. The approach follows the three guiding

principles listed above and, in particular, ensures that every rider has a guaranteed ride back. The contributions of this study are threefold:

1. It first defines the CTSP to formally capture the matching problem described previously.
2. It proposes two algorithms, an REA and a Branch-and-Price Algorithm (BPA), to solve the CTSP for a cluster of riders.
3. It analyzes the scalability of the two algorithms along different dimensions, including the capacity of the vehicles, the size of the clusters, and their ability to be deployed in real situations.

The CTSP can be viewed as a generalization of the VRP with routes satisfying time-window, capacity, pairing, precedence, ride-duration, and driver constraints. In addition to picking up and dropping off riders within desired time windows while ensuring vehicle capacities are not exceeded, routes in the CTSP must also ensure their ride durations are not excessively long to limit user inconvenience. In this sense, the CTSP shares some similarities with the DARP. It differs from the DARP in that it relies on the use of personal vehicles to serve all trip requests, which come in pairs for each commuter as each rider makes a trip to the workplace and another back home. The drivers of these vehicles therefore belong the set of riders, and their routes to the workplace and back home must be carefully constructed and balanced to ensure that every rider is covered on their way to work and guaranteed a ride back home. These additional requirements make the CTSP unique and particularly challenging. The CTSP also uses a lexicographic objective function that first minimizes the number of cars and then the total travel distance.

This study proposes two exact algorithms for the CTSP: An REA which exhaustively searches for feasible routes from all possible trip combinations before route selection is optimized with a MIP, and a BPA which uses column generation and a pricing algorithm based on dynamic programming. On top of the two algorithms, this study highlights the key characteristics of the CTSP that differentiate it from the DARP that allow its routes to be enumerated by the REA. While the BPA builds on conventional techniques to solve the CTSP via column generation, it introduces a wait-time relaxation technique, which is a novel alternative to the weak and strong dominance relations proposed by Gschwind and Irnich (2015) for finding feasible routes that simultaneously satisfy time-window and ride-duration constraints in the pricing problem. This study also proposes a time-limited, root-node heuristic which is derived from the BPA and demonstrates its capability to produce high-quality solutions for medium to large problem instances within a 10-minute time span, *making it well suited for time-constrained scenarios within an operational setting*. Finally, this study proposes and compares a couple of clustering algorithms to decompose large-

scale problems by grouping commuters based on their home locations and trip schedules. The algorithms are then used to generate problem instances from the real-world dataset of commute trips from the city of Ann Arbor, Michigan.

The remainder of this chapter is organized as follows. Section 3.2 provides a formal definition and a mathematical formulation of the CTSP, followed by Section 3.3 which discusses the first algorithm to solve the problem, the REA. Next, Section 3.4 describes the second algorithm, the BPA, together with its derived root-node heuristic. The clustering algorithms are presented in Section 3.5, while the computational results are reported in Section 3.6. Finally, some concluding remarks are provided in Section 3.7.

## 3.2 The Commute Trip-Sharing Problem

The CTSP aims at finding a set of minimum-cost feasible routes to cover all inbound and outbound trips of a set of commuters $\mathcal{C}$ for a given day while ensuring the set of drivers for the inbound and the outbound routes are identical. Let $\Omega^+$ and $\Omega^-$ denote the set of all feasible inbound and outbound routes respectively, and $c_r$ denote the cost of route $r$. The CTSP formulation uses a binary variable $X_r$ to indicate whether a route $r \in \Omega^+ \cup \Omega^-$ is selected, a binary constant $\alpha_{r,i}$ which is equal to 1 iff route $r$ serves rider $i$ (i.e., $\alpha_{r,i} = 1$ iff $i \in \mathcal{C}_r$), and a binary constant $\beta_{r,i}$ which is equal to 1 iff rider $i$ is the driver of route $r$ (i.e., $\beta_{r,i} = 1$ iff $i = D_r$). The problem formulation is given by (3.1)–(3.5).

$$\min \quad \sum_{r \in \Omega^+ \cup \Omega^-} c_r X_r \tag{3.1}$$

$$\text{s.t.} \quad \sum_{r \in \Omega^+} \alpha_{r,i} X_r = 1 \qquad \forall i \in \mathcal{C} \tag{3.2}$$

$$\sum_{r \in \Omega^-} \alpha_{r,i} X_r = 1 \qquad \forall i \in \mathcal{C} \tag{3.3}$$

$$\sum_{r \in \Omega^+} \beta_{r,i} X_r - \sum_{\hat{r} \in \Omega^-} \beta_{\hat{r},i} X_{\hat{r}} = 0 \qquad \forall i \in \mathcal{C} \tag{3.4}$$

$$X_r \in \{0,1\} \qquad \forall r \in \Omega^+ \cup \Omega^- \tag{3.5}$$

The model features a lexicographic objective that first minimizes the number of vehicles and then the total distance. It is rewritten into a single objective by appropriate weighting of the two sub-objectives. The cost $c_r$ penalizes the total distance of route $r$ and heavily penalizes its selection. Let $\varsigma_{(i,j)}$ denote the *distance* of the shortest path between nodes $i$

and $j$. The cost $c_r$ is then given by the addition of variable and fixed costs of the route:

$$c_r = \hat{c}_r + \bar{c} \tag{3.6}$$

where the variable and fixed costs, $\hat{c}_r$ and $\bar{c}$, are given by:

$$\hat{c}_r = \sum_{(i,j)\in r} \varsigma_{(i,j)} \tag{3.7}$$

$$\bar{c} = M \max_{r\in\Omega^+\cup\Omega^-} \sum_{(i,j)\in r} \varsigma_{(i,j)} \tag{3.8}$$

where $M$ is a large number. In practice, $M$ is set to 1000, which is sufficiently large to ensure that the number of selected routes is first minimized followed by their total distance. Constraints (3.2) and (3.3) enforce coverage of each rider's inbound and outbound trips by exactly one route each, while constraints (3.4) ensure drivers of inbound and outbound routes are identical. The set-partitioning problem of (3.1)–(3.5) is referred to as the MP throughout the rest of this chapter.

The CTSP is essentially a VRP with capacity, time-window, pairing, precedence, ride-duration, and driver constraints, making it most similar to the DARP. However, the key distinctions of the CTSP are:

(a) Drivers in the CTSP are members of the set of riders, i.e., $D_r \in \mathcal{C}_r$. This leads to driver constraints which require routes to start and end at the drivers' origins and destinations respectively, whereas requests in the DARP are served by shared vehicles whose routes begin and end at a central depot.

(b) The set of drivers for the inbound and the outbound routes needs to be balanced, leading to constraints (3.4) in the MP. These constraints add another layer of complexity which is not present in the DARP.

Therefore, the CTSP can also be seen as a DARP with additional constraints.

## 3.3 The Route-Enumeration Algorithm

The first approach for solving the CTSP is by enumerating all routes in $\Omega^+ \cup \Omega^-$ before solving the MP with a MIP solver. The REA, described in detail in Section 2.2.2, supports this approach by exhaustively searching for these routes from all possible combinations of inbound or outbound trips with a maximum cardinality of $K$. Therefore, in this approach, the REA is first executed to enumerate $\Omega^+ \cup \Omega^-$, after which the MP is solved to obtain a solution to the CTSP.

The fact that the drivers are commuters themselves is the key characteristic of the CTSP; it allows its routes to be exhaustively enumerated. Indeed, drivers must complete their trips within specific time windows and, most importantly, their trips are subject to ride-duration constraints. As a result, in general, a route typically consists of three phases: A pickup phase where the driver picks up passengers, a driving phase where the vehicle travels to the destination, and a drop-off phase where the driver drops off all the passengers before ending her trip. After the drop-offs, the driver has no time to go back and pick up another set of passengers due to her trip's time-window and ride-duration constraints. This permits the REA to consider only routes that contain up to $K$ passengers to enumerate all possible routes, and $K$ is typically small. In fact, as shown in Section 2.2.2, $|\,\Omega^+ \cup \Omega^-\,| = \mathcal{O}(n^K)$. In contrast, the DARP uses dedicated drivers who are not subjected to any ride-duration constraints and can serve riders throughout the day. Therefore it cannot restrict its attention to routes with only $K$ passengers, as the number of passengers in its routes is not limited by the capacity of the vehicle, but it is limited by the total number of travelers.

## 3.4 The Branch-and-Price Algorithm

The BPA combines existing techniques with some novel elements to solve the CTSP. At its core is a conventional column-generation algorithm which utilizes a RMP—the linear relaxation of the MP defined on a subset of all feasible routes $\Omega^{+\prime} \cup \Omega^{-\prime}$—and solves a PSP to identify new feasible routes with negative reduced costs to augment $\Omega^{+\prime} \cup \Omega^{-\prime}$. The PSP solves several dynamic programs that search for resource-constrained shortest paths representing the feasible routes. The column-generation algorithm solves the RMP and the PSP iteratively until the PSP is unable to find any routes with negative reduce cost, at which point the objective value of the RMP converges to the optimal objective value of the linear relaxation of the original MP. Furthermore, the solution of the RMP represents the optimal solution to the original MP *if it is integral* at the convergence stage. Otherwise, a bi-level branching strategy that is tailored specifically for the CTSP is employed to search for the optimal integer solution.

*This work introduces a novel wait-time relaxation technique that not only obtains feasible routes that simultaneously satisfy time-window and ride-duration constraints in the PSP, but also guarantees elementarity of the routes.* It proposes utilization of a resource that models trip durations excluding wait times which allow the dynamic programs to produce preliminary routes with minimal reduced costs that first satisfy a set of constraints necessary for route feasibility. The feasibility of the preliminary routes are then evaluated with the inclusion of wait times, and infeasible ones are added to a set of forbidden paths

whose members are prevented from subsequent discovery via the dynamic-programming approach of Di Puglia Pugliese and Guerriero (2013a).

### 3.4.1 The Pricing Subproblem

The PSP is responsible for finding new feasible routes with negative reduced costs. Letting $\pi_i^+$, $\pi_i^-$, and $\sigma_i$ denote the optimal duals of constraints (3.2), (3.3), and (3.4) of the RMP respectively, the reduced cost of an inbound route $r^+$ is given by:

$$rc_{r^+} = c_{r^+} - \sum_{i \in \mathcal{C}_{r^+}} \pi_i^+ - \sigma_{D_{r^+}} \tag{3.9}$$

while that of an outbound route $r^-$ is given by:

$$rc_{r^-} = c_{r^-} - \sum_{i \in \mathcal{C}_{r^-}} \pi_i^- + \sigma_{D_{r^-}} \tag{3.10}$$

The desired routes are obtained by considering each rider $d \in \mathcal{C}$ as the driver of an inbound route $r_d^+$ and an outbound route $r_d^-$, and then finding such routes with minimum reduced costs. To obtain these routes, the algorithm builds a pair of graphs $\mathcal{G}_d^+$ and $\mathcal{G}_d^-$ for each $d \in \mathcal{C}$. In the following, $\mathcal{G}$ denotes the set of all constructed graphs, i.e., $\mathcal{G} = \{\mathcal{G}_d^+ : d \in \mathcal{C}\} \cup \{\mathcal{G}_d^- : d \in \mathcal{C}\}$. Without loss of generality, the presentation outlines how a route $r_d^+$ with minimal reduced cost is found from $\mathcal{G}_d^+$.

First, let $n = |\,\mathcal{C}\,|$, and $\mathcal{O}^+ = \{1, \cdots, n\}$ and $\mathcal{D}^+ = \{n+1, \cdots, 2n\}$ denote the sets of all origin and destination nodes respectively. The origin and destination of rider $i$ are then represented by nodes $i$ and $n+i$ respectively. The graph $\mathcal{G}_d^+ = (\mathcal{N}_d^+, \mathcal{A}_d^+)$ is built with nodes $\mathcal{N}_d^+ = \mathcal{O}^+ \cup \mathcal{D}^+$ and fully-connected edges $\mathcal{A}_d^+$. A ride-duration limit $L_i$ and a demand $\kappa_i$, representing the number of riders to be picked up at node $i$, are then associated with each node $i \in \mathcal{O}^+$, a time window $[a_i, b_i]$ and a service duration $\zeta_i$ are associated with each node $i \in \mathcal{N}_d^+$, and a travel time $\tau_{(i,j)}$ and a reduced cost $c_{(i,j)}$ are associated with each edge $(i,j) \in \mathcal{A}_d^+$. Letting $\delta^+(i)$ and $\delta^-(i)$ denote the set of outgoing and incoming edges of node $i$, the edge costs are defined as follows so that the total cost of any path from $d$ to $n+d$ is equivalent to $rc_{r_d^+}$:

$$c_{(i,j)} = \begin{cases} \bar{c} + \varsigma_{(i,j)} - \pi_i^+ - \sigma_d & \forall (i,j) \in \delta^+(d) \\ \varsigma_{(i,j)} - \pi_i^+ & \forall i \in \mathcal{O}^+ \setminus \{d\},\ \forall (i,j) \in \delta^+(i) \\ \varsigma_{(i,j)} & \forall i \in \mathcal{D}^+,\ \forall (i,j) \in \delta^+(i) \end{cases} \tag{3.11}$$

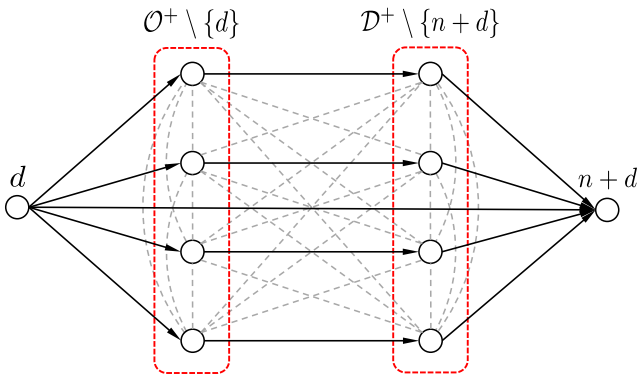



Figure 3.2: Graph $\mathcal{G}_d^+$ After Application of Edge Elimination Rules (a) and (b) from Section 3.4.2 (Each Dotted Line Represents a Pair of Bidirectional Edges).

Similarly, letting $\mathcal{O}^-$ and $\mathcal{D}^-$ denote the sets of all outbound origin and destination nodes, the graph $\mathcal{G}_d^- = (\mathcal{N}_d^-, \mathcal{A}_d^-)$ is built with nodes $\mathcal{N}_d^- = \mathcal{O}^- \cup \mathcal{D}^-$ and fully-connected edges $\mathcal{A}_d^-$, and the costs of edges $(i,j) \in \mathcal{A}_d^-$ are defined as follows to ensure the total cost of any path from $d$ to $n+d$ in $\mathcal{G}_d^-$ is equal to $rc_{r_d^-}$:

$$c_{(i,j)} = \begin{cases} \bar{c} + \varsigma_{(i,j)} - \pi_i^- + \sigma_d & \forall (i,j) \in \delta^+(d) \\ \varsigma_{(i,j)} - \pi_i^- & \forall i \in \mathcal{O}^- \setminus \{d\},\ \forall (i,j) \in \delta^+(i) \\ \varsigma_{(i,j)} & \forall i \in \mathcal{D}^-,\ \forall (i,j) \in \delta^+(i) \end{cases} \tag{3.12}$$

A priori feasibility constraints, further detailed in Section 3.4.2, are then applied to identify and eliminate edges that cannot belong to any feasible route. Figure 3.2 provides a sketch of $\mathcal{G}_d^+$ after application of several of these edge-elimination rules.

The minimum-reduced-cost $r_d^+$ is then obtained by finding the least-cost feasible path from $d$ to $n+d$ in $\mathcal{G}_d^+$. Recall that for the path to be feasible, it must satisfy the time-window, capacity, pairing, precedence, ride-duration, and driver constraints. The problem is therefore an ESPPRC which is known to be NP-hard (Dror 1994). While the driver constraint is enforced by construction by making $d$ the source and $n+d$ the target of the shortest-path problem, the remaining constraints are implemented by introducing and enforcing constrained resources in a Resource-Constrained Shortest Path Algorithm (RCSPA) which is further elaborated in Section 3.4.3. On the whole, the PSP involves solving $2n$ independent ESPPRCs to produce up to $2n$ feasible routes with negative reduced costs.

### 3.4.2 Time-Window Tightening and Edge Elimination

Pre-processing of the time-window, precedence, pairing, capacity, ride-duration limit, and driver constraints makes it possible to identify edges that cannot belong to any feasible route which may then be removed from $\mathcal{G}$. Without loss of generality, the following description focuses on edge elimination for $\mathcal{G}_d^+$.

Prior to determining infeasible edges, the time windows of all nodes are tightened by sequentially reducing their upper and lower bounds using the following rules introduced by Dumas et al. (1991).

- $b_i = \min\{b_i, b_{n+d} - \zeta_i - \tau_{(i,n+d)}\}$, $\forall i \in \mathcal{D}^+ \setminus \{n+d\}$
- $b_i = \min\{b_i, b_{n+i} - \zeta_i - \tau_{(i,n+i)}\}$, $\forall i \in \mathcal{O}^+ \setminus \{d\}$
- $a_i = \max\{a_i, a_d + \zeta_d + \tau_{(d,i)}\}$, $\forall i \in \mathcal{O}^+ \setminus \{d\}$
- $a_i = \max\{a_i, a_{i-n} + \zeta_{i-n} + \tau_{(i-n,i)}\}$, $\forall i \in \mathcal{D}^+ \setminus \{n+d\}$

The following constraints and rules, derived by combining those proposed by Dumas et al. (1991) and Cordeau (2006), are then applied to identify and eliminate infeasible edges:

(a) Driver: Edges $\{(d, n+i), (i, d), (i, n+d), (n+i, d), (n+d, i), (n+d, n+i) : i \in \mathcal{O}^+ \setminus \{d\}\}$.

(b) Pairing and precedence: Edges $\{(n+i, i) : i \in \mathcal{O}^+\}$.

(c) Capacity: Edges $\{(i,j), (j,i), (i,n+j), (j,n+i), (n+i,n+j), (n+j,n+i) : i,j \in \mathcal{O}^+ \wedge i \neq j \wedge \kappa_i + \kappa_j > K\}$.

(d) Time windows: Edges $\{(i,j) : (i,j) \in \mathcal{A}_d^+ \wedge a_i + \zeta_i + \tau_{(i,j)} > b_j\}$.

(e) Ride-duration limit: Edges $\{(i,j), (j,n+i) : i \in \mathcal{O}^+ \wedge j \in \mathcal{N}_d^+ \wedge i \neq j \wedge \tau_{(i,j)} + \zeta_j + \tau_{(j,n+i)} > L_i\}$.

(f) Pairing, time windows, and ride-duration limit:

- Edges $\{(i, n+j) : i,j \in \mathcal{O}^+ \wedge i \neq j \wedge \neg feasible(j \to i \to n+j \to n+i)\}$.
- Edges $\{(n+i, j) : i,j \in \mathcal{O}^+ \wedge i \neq j \wedge \neg feasible(i \to n+i \to j \to n+j)\}$.
- Edges $\{(i,j) : i,j \in \mathcal{O}^+ \wedge i \neq j \wedge \neg feasible(i \to j \to n+i \to n+j) \wedge \neg feasible(i \to j \to n+j \to n+i)\}$.
- Edges $\{(n+i, n+j) : i,j \in \mathcal{O}^+ \wedge i \neq j \wedge \neg feasible(i \to j \to n+i \to n+j) \wedge \neg feasible(j \to i \to n+i \to n+j)\}$.

Note that the rules in (f) utilize the $feasible$ function introduced in Section 1.1.2 to determine if a partial route satisfies the time-window and the ride-duration limit constraints. For instance, the first says edge $(i, n+j)$ is infeasible if route $j \to i \to n+j \to n+i$ is infeasible. Edge elimination rules for $\mathcal{G}_d^-$ are obtained by replacing $\mathcal{O}^+$, $\mathcal{D}^+$, and $\mathcal{A}_d^+$ in the rules above with $\mathcal{O}^-$, $\mathcal{D}^-$, and $\mathcal{A}_d^-$ respectively.

### 3.4.3 The Resource-Constrained Shortest Path Algorithm

The PSP uses an RCSPA that is based on the label-setting dynamic program proposed by Desrochers (1988) to find the least-cost feasible path from any graph in $\mathcal{G}$, i.e., one that satisfies time-window, capacity, pairing, precedence, and ride-duration constraints. The path searched by this algorithm is identical to that sought in the PSP of the DARP by Gschwind and Irnich (2015). Their method incorporated novel dominance rules in the labeling procedure to directly enforce all constraints in the dynamic program.

On the other hand, the RCSPA presented here first searches for the *minimum-cost, feasible route that ignores the wait times.* The routes that are infeasible with respect to the wait times are then pruned in a second step. This procedure is motivated by the fact that the optimal values for the wait times require knowledge of the complete route, which is only known at the end of the search. By relaxing the wait times, the dynamic program first finds a candidate route which is later evaluated for feasibility with respect to the wait times once it is complete. Moreover, subsequent empirical evaluations revealed that for the problem instances considered, an overwhelming majority of the candidate routes are feasible with the inclusion of wait times. The resources utilized in the algorithm are also capable of guaranteeing generation of elementary paths.

The RCSPA can therefore be seen as a middle ground approach between the method by Ropke and Cordeau (2006) which completely relaxes the ride-duration constraint in the PSP and prevents selection of paths that violate the constraint through infeasible path elimination constraints in the RMP, and that by Gschwind and Irnich (2015) which directly enforces all constraints in the dynamic program of the PSP. Without loss of generality, this section describes the algorithm for $\mathcal{G}_d^+$.

#### 3.4.3.1 Label definition

Let $\mathcal{P}_l^k$ denote the $k^{\text{th}}$ path from the source $d$ to node $l$. A label $\mathcal{L}_l^k$ with five resources $(c_l^k, T_l^k, \mathcal{C}_l^k, \mathcal{R}_l^k, \mathcal{W}_l^k)$ is associated with each $\mathcal{P}_l^k$. Resource $c_l^k$ represents the total cost of edges in $\mathcal{P}_l^k$, i.e., $c_l^k = \sum_{(i,j)\in\mathcal{P}_l^k} c_{(i,j)}$, whereas $T_l^k$ is the time at which service at node $l$ begins for $\mathcal{P}_l^k$. Resource $\mathcal{C}_l^k$ denotes the set of riders on the vehicle right after visiting node $l$ on $\mathcal{P}_l^k$. It is equivalent to the set of pickup nodes visited on $\mathcal{P}_l^k$ whose corresponding drop-off nodes have yet to be visited. On the other hand, $\mathcal{R}_l^k$ denotes the set of all riders that have been picked up by $\mathcal{P}_l^k$ after visiting node $l$. It is equivalent to the set of all pickup nodes visited by $\mathcal{P}_l^k$. Finally, $\mathcal{W}_l^k$ is the set of trip durations, *excluding wait times*, for each rider in $\mathcal{R}_l^k$. Letting $\mathcal{P}_l^k(m)$ denote the set of edges from $\mathcal{P}_l^k$ on which rider $m$ is on the vehicle and $w_l^k(m)$ be the trip duration of rider $m$ excluding wait times on $\mathcal{P}_l^k$, i.e.,

$w_l^k(m) = \sum_{(i,j)\in\mathcal{P}_l^k(m)} \zeta_i + \tau_{(i,j)}$, then $\mathcal{W}_l^k = \{w_l^k(m) : m \in \mathcal{R}_l^k\}$. The load $Y_l^k$ of a vehicle after visiting node $l$ on path $\mathcal{P}_l^k$ can be easily obtained from $Y_l^k = \sum_{i\in\mathcal{C}_l^k} \kappa_i$. Therefore, $\mathcal{L}_l^k$ contains sufficient information to ensure $\mathcal{P}_l^k$ satisfies pairing, precedence, time-window, and capacity constraints. While resource $\mathcal{W}_l^k$ is not sufficient for verifying compliance to the ride-duration limit for each rider, it does provide a lower bound to each ride duration which must necessarily satisfy the limit for $\mathcal{P}_l^k$ to be feasible.

#### 3.4.3.2 Label extension

Label $\mathcal{L}_l^k$ is maintained using a forward dynamic program. In the label-setting algorithm, an attempt is made to extend $\mathcal{L}_l^k$ along edge $(l, j)$ to produce label $\mathcal{L}_j^{k'}$ for path $\mathcal{P}_j^{k'}$. The resources in $\mathcal{L}_j^{k'}$ are calculated as follows:

$$c_j^{k'} = c_l^k + c_{(l,j)} \tag{3.13}$$

$$T_j^{k'} = \begin{cases} \max\{a_j, T_l^k + \zeta_l + \tau_{(l,j)}\} & \text{if } j \in \mathcal{O}^+ \\ T_l^k + \zeta_l + \tau_{(l,j)} & \text{otherwise} \end{cases} \tag{3.14}$$

$$\mathcal{C}_j^{k'} = \begin{cases} \mathcal{C}_l^k \cup \{j\} & \text{if } j \in \mathcal{O}^+ \\ \mathcal{C}_l^k \setminus \{j-n\} & \text{otherwise} \end{cases} \tag{3.15}$$

$$\mathcal{R}_j^{k'} = \mathcal{R}_l^k \cup \{j\} \qquad \text{if } j \in \mathcal{O}^+ \tag{3.16}$$

$$w_j^{k'}(j) = 0 \qquad \text{if } j \in \mathcal{O}^+ \wedge j \notin \mathcal{R}_l^k \tag{3.17}$$

$$w_j^{k'}(i) = w_l^k(i) + \zeta_l + \tau_{(l,j)} \qquad \forall i \in \mathcal{C}_l^k \tag{3.18}$$

The extension is performed if and only if:

$$T_j^{k'} \le b_j, \tag{3.19}$$

$$j \notin \mathcal{C}_l^k \qquad \text{if } j \in \mathcal{O}^+, \tag{3.20}$$

$$j - n \in \mathcal{C}_l^k \qquad \text{if } j \in \mathcal{D}^+, \tag{3.21}$$

$$\sum_{i\in\mathcal{C}_j^{k'}} \kappa_i \le K, \text{ and} \tag{3.22}$$

$$w_j^{k'}(i) - \zeta_i \le L_i \qquad \forall i \in \mathcal{C}_l^k. \tag{3.23}$$

Constraints (3.19)–(3.23) list conditions that are necessary to ensure feasibility of $\mathcal{P}_j^{k'}$. Note that if $w_j^{k'}(i) - \zeta_i$, which constitutes rider $i$'s ride duration excluding wait times and hence is the lower bound to her ride duration, is already exceeding $L_i$, then $L_i$ will certainly be exceeded if wait times were included. Therefore, the conditions in (3.23) are necessary but

not sufficient for enforcing the ride-duration limit constraint for each rider.

The algorithm is initialized by path $\mathcal{P}_d^1$ whose label $\mathcal{L}_d^1 = (0, a_d, \{d\}, \{d\}, \{0\})$, and a preliminary solution is given by path $\mathcal{P}_{n+d}^{k^*}$ whose cost $c_{n+d}^{k^*}$ is minimal and whose resource $\mathcal{C}_{n+d}^{k^*} = \emptyset$. Note that a non-elementary path may result if the graph contains a negative-cost cycle. However, such paths may be eliminated by setting the ride-duration limit of each rider to be less than twice the ride duration of her direct trip, i.e., $L_i < 2\tau_{(i,n+i)} + \zeta_i$.

**Proposition 3.4.1.** Non-elementary paths will not be generated by the RCSPA if $L_i < 2\tau_{(i,n+i)} + \zeta_i$ for each $i \in \mathcal{O}^+$.

*Proof.* Suppose a non-elementary path is generated by the RCSPA. On the path, there must exist at least one rider $i$ who is served more than once. For such riders, both $i$ and $n+i$ must be visited more than once with $i$ preceding $n+i$ each time and $n+i$ being visited first before $i$ is visited again due to the pairing and precedence constraints. As a result, resource $w_{n+d}^{k^*}(i) \geq 2(\zeta_i + \tau_{(i,n+i)})$ and therefore $w_{n+d}^{k^*}(i) - \zeta_i \geq 2\tau_{(i,n+i)} + \zeta_i$. If $L_i < 2\tau_{(i,n+i)} + \zeta_i$, then $w_{n+d}^{k^*}(i) - \zeta_i > L_i$. Condition (3.23) is thus violated, causing the path to not be extended. □

Also note that as the restrictions on $\mathcal{W}_l^k$ are not sufficient for ensuring satisfaction of the ride-duration constraints, $\mathcal{P}_{n+d}^{k^*}$ may be infeasible. Therefore, an additional step needs to be performed to verify the feasibility of $\mathcal{P}_{n+d}^{k^*}$.

#### 3.4.3.3 Forbidding paths violating the ride-duration limit

Feasibility of the preliminary solution $\mathcal{P}_{n+d}^{k^*}$ with the inclusion of wait times can be verified using the $feasible$ function as $\mathcal{P}_{n+d}^{k^*}$ represents a complete route. A feasible path $\mathcal{P}_{n+d}^{k^*}$ represents the optimal solution to the ESPPRC. While initial empirical evaluations revealed that the vast majority of the preliminary routes found ($> 99\%$ of the paths found) are feasible, infeasible paths are still discovered on rare occasions. In such cases, the infeasible path is added to a set of forbidden paths associated with the graph, after which the RCSPA is executed again repeatedly to generate newer paths until a feasible one is found.

The shortest path problem with forbidden paths (Villeneuve and Desaulniers 2005, Di Puglia Pugliese and Guerriero 2013b,a) is a method that has been successfully applied for handling constraints which are hard or impossible to model as resources. This work exploits this idea to properly enforce the ride-duration limit constraints by preventing infeasible preliminary routes from being discovered by the RCSPA again. The dynamic-programming approach of Di Puglia Pugliese and Guerriero (2013a) is employed for this purpose since it fits well into the label-setting framework.

Firstly, let $\mathcal{F}_d^+$ denote the set of forbidden paths for $\mathcal{G}_d^+$. Also let $\dot{f}$ denote the first edge of a forbidden path $f$, $\mid f \mid$ denote the total number of edges on the path, and $h_l^k(f)$ denote the number of consecutive edges of $f$ starting from $\dot{f}$ that is present in $\mathcal{P}_l^k$. To forbid paths in $\mathcal{F}_d^+$ from being discovered by the RCSPA, an additional resource $\mathcal{H}_l^k = \{h_l^k(f) : f \in \mathcal{F}_d^+\}$ is introduced to the label so that $\mathcal{L}_l^k = (c_l^k, T_l^k, \mathcal{C}_l^k, \mathcal{R}_l^k, \mathcal{W}_l^k, \mathcal{H}_l^k)$. During label extension along edge $(l, j)$, $\mathcal{H}_j^{k'}$ is calculated as follows:

$$h_j^{k'}(f) = \begin{cases} 1 & \text{if } (l,j) \in f \wedge h_l^k(f) = 0 \wedge (l,j) = \dot{f} \\ 0 & \text{if } (l,j) \in f \wedge h_l^k(f) = 0 \wedge (l,j) \neq \dot{f} \\ h_l^k(f) + 1 & \text{if } (l,j) \in f \wedge h_l^k(f) \geq 1 \wedge consec(\mathcal{P}_l^k, (l,j), f) \\ 0 & \text{if } (l,j) \in f \wedge h_l^k(f) \geq 1 \wedge \neg consec(\mathcal{P}_l^k, (l,j), f) \\ 0 & \text{if } (l,j) \notin f \end{cases} \qquad \forall f \in \mathcal{F}_d^+ \tag{3.24}$$

$consec(\mathcal{P}_l^k, (l,j), f)$ is a function that returns true if there exists a set of consecutive edges in path $\{\mathcal{P}_l^k, (l,j)\}$ ending with $(l,j)$ that exactly matches a set of consecutive edges in path $f$ starting from $\dot{f}$, and returns false otherwise. The extended resource must then satisfy the following constraints:

$$h_j^{k'}(f) \leq \mid f \mid - 1 \qquad \forall f \in \mathcal{F}_d^+ \tag{3.25}$$

since $\mathcal{P}_j^{k'}$ would contain a forbidden path otherwise. The resource is initialized with $h_d^1(f) = 0$ for each $f \in \mathcal{F}_d^+$. Resource $\mathcal{H}_l^k$ prevents the RCSPA from discovering infeasible preliminary routes stored in $\mathcal{F}_d^+$ again, thus ensuring that the algorithm's solution is always feasible.

#### 3.4.3.4 Label elimination

As efficiency of the label-setting algorithm increases with the number of eliminated labels, a label and its associated path is eliminated if it is established that they cannot belong to either an optimal or a feasible solution. Firstly, dominance rules are applied to determine if a label does not belong to an optimal solution.

**Definition 3.4.1** (Label Domination)**.** $\mathcal{L}_l^k$ dominates $\mathcal{L}_l^{k'}$ if and only if:

$$c_l^k \leq c_l^{k'}, \tag{3.26}$$
$$T_l^k \leq T_l^{k'}, \tag{3.27}$$
$$\mathcal{C}_l^k \subseteq \mathcal{C}_l^{k'}, \tag{3.28}$$
$$w_l^k(i) \leq w_l^{k'}(i) \qquad \forall i \in \mathcal{C}_l^k, \text{ and} \tag{3.29}$$
$$h_l^k(f) \leq h_l^{k'}(f) \qquad \forall f \in \mathcal{F}_d^+. \tag{3.30}$$

If $\mathcal{L}_l^{k'}$ is dominated by $\mathcal{L}_l^k$, then $\mathcal{L}_l^{k'}$ and its associated path $\mathcal{P}_l^{k'}$ cannot belong to an optimal solution to the ESPPRC as every feasible extension to $\mathcal{P}_l^{k'}$ is also applicable to $\mathcal{P}_l^k$ at an equal or lower cost. Therefore $\mathcal{L}_l^{k'}$ and $\mathcal{P}_l^{k'}$ may be eliminated.

Next, the following rules are applied to identify labels that cannot belong to a feasible solution:

(a) $\mathcal{L}_l^k$ such that $\mathcal{C}_l^k \setminus \{d\} \neq \emptyset$ is eliminated if there exists $i \in \mathcal{C}_l^k \setminus \{d\}$ where the path extension $l \to n+i \to n+d$ is infeasible.

(b) $\mathcal{L}_l^k$ such that $|\, \mathcal{C}_l^k \setminus \{d\} \,| \geq 2$ is eliminated if there exists $i, j \in \mathcal{C}_l^k \setminus \{d\} \wedge i \neq j$ where path extensions $l \to n+i \to n+j \to n+d$ and $l \to n+j \to n+i \to n+d$ are both infeasible.

For the rules above, feasibility of the path extensions are verified by checking if they satisfy the time-window and ride-duration constraints excluding wait times, i.e., by checking if each node along the extension satisfies conditions (3.19) and (3.23). The rules are inspired by the notion of non-post-feasible labels introduced by Dumas et al. (1991). They are essentially heuristics which check if at least one (for rule (a)) or two (for rule (b)) of the riders on the vehicle, excluding the driver, can be delivered to their destinations while respecting their time windows and ride-duration limits if wait times are ignored. While not sufficient, these conditions are necessary for the feasibility of any extension to $\mathcal{P}_l^k$, and they result in the elimination of a large number of infeasible labels in practice.

### 3.4.4 Obtaining an Integer Solution

The unique structure of the MP lets us infer a few properties about its solution. Firstly, since the total number of selected inbound routes must match that of outbound routes in any solution, the total number of selected routes in an integer solution must be even, i.e., $\sum_{r \in \Omega^+ \cup \Omega^-} X_r \in \{2a : a \in \mathbb{Z}_{\geq 0}\}$. Secondly, since only integral distances are used in this work, all routes costs and consequently the objective value of an integer solution must also be integral, i.e., $\sum_{r \in \Omega^+ \cup \Omega^-} c_r X_r \in \mathbb{Z}_{\geq 0}$.

These two properties are leveraged to obtain an integer solution should the optimal solution of the RMP not be integral. Let $\chi^*$ denote the total number of selected routes at column-generation convergence, i.e., $\chi^* = \sum_{r \in \Omega^{+\prime} \cup \Omega^{-\prime}} X_r$, and $z^*$ denote the objective value at convergence. If $\chi^*$ is not an even integer, the following cut is introduced to the RMP to round up the total number of selected routes to the nearest even integer.

$$\sum_{r \in \Omega^{+\prime} \cup \Omega^{-\prime}} X_r \geq 2 \left\lceil \frac{\chi^*}{2} \right\rceil \tag{3.31}$$

The dual of the cut is appropriately transferred to the PSP and the column-generation procedure is resumed until convergence again. If $z^*$ is not integral at this point, another cut is added to the RMP to round up its objective value to the nearest integer:

$$\sum_{r \in \Omega^{+\prime} \cup \Omega^{-\prime}} c_r X_r \geq \lceil z^* \rceil \tag{3.32}$$

Once again, the dual of the cut is transferred to the PSP and the column-generation procedure is resumed until convergence. If the solution of the RMP is still not integral at this stage, then a branch-and-bound tree needs to be explored whereby additional columns may be generated at each tree node.

A bi-level branching scheme is employed for the branch-and-bound tree, whereby integrality of driver selection is enforced in the first level and integrality of edge flow is enforced in the second. In the first level, let $V_i$ be a variable that indicates whether rider $i$ is selected as the driver in a solution. It is given by:

$$V_i = \sum_{r \in \Omega^{+\prime}} \beta_{r,i} X_r \qquad \forall i \in \mathcal{C} \tag{3.33}$$

In an integral solution, all $V_i$s must be binary. Therefore if they are not, a fractional $V_i$ is selected and two branches are created; one fixing it to 0 and another fixing it to 1. The branch decision of $V_i = 0$ is enforced in the RMP by removing columns where rider $i$ is the driver, i.e., $\{r \in \Omega^{+\prime} \cup \Omega^{-\prime} : D_r = i\}$, while it is enforced in the PSP by not solving the ESPPRC on graphs where rider $i$ is the driver, i.e., $\mathcal{G}_i^+$ and $\mathcal{G}_i^-$. To enforce $V_i = 1$, the following cut is introduced to the RMP:

$$\sum_{r \in \Omega^{+\prime} : D_r = i} X_r = 1 \tag{3.34}$$

while ensuring its dual is properly incorporated into the PSP. No additional steps are

needed to enforce the branch decision in the PSP since the ESPPRCs on graphs $\mathcal{G}_i^+$ and $\mathcal{G}_i^-$ are already being solved by default.

If all $V_i$s are binary and the solution of the RMP is still fractional, then a second branching scheme based on that proposed by Desrochers et al. (1992) is utilized. In the second level, let $\omega(i,j)$ denote the set of all routes utilizing edge $(i,j)$, i.e., $\omega(i,j) = \{r \in \Omega^{+\prime} \cup \Omega^{-\prime} : (i,j) \in r\}$, and let $F_{(i,j)}$ be the flow variable for edge $(i,j)$ that indicates if node $i$ should be served before node $j$ in a solution. It is given by:

$$F_{(i,j)} = \sum_{r \in \omega(i,j)} X_r \qquad \forall (i,j) \in \{\mathcal{A}_d^+ \cup \mathcal{A}_d^- : d \in \mathcal{C}\} \tag{3.35}$$

Also let $\mathcal{A}^+$ and $\mathcal{A}^-$ denote the set of edges from all inbound and outbound graphs respectively, i.e., $\mathcal{A}^+ = \{\mathcal{A}_d^+ : d \in \mathcal{C}\}$, $\mathcal{A}^- = \{\mathcal{A}_d^- : d \in \mathcal{C}\}$. In an integer solution, all $F_{(i,j)}$s must be binary. In a fractional solution however, one of the following cases may occur:

(a) $F_{(i,j)}$ for all $(i,j) \in \mathcal{A}^+$ are binary, but there exists $(u,v) \in \mathcal{A}^-$ such that $F_{(u,v)}$ is fractional.

(b) $F_{(u,v)}$ for all $(u,v) \in \mathcal{A}^-$ are binary, but there exists $(i,j) \in \mathcal{A}^+$ such that $F_{(i,j)}$ is fractional.

(c) There exist $(i,j) \in \mathcal{A}^+$ and $(u,v) \in \mathcal{A}^-$ such that both $F_{(i,j)}$ and $F_{(u,v)}$ are fractional.

If either case (a) or (b) occurs, then an edge $(i,j)$ whose flow is fractional is selected (from either $\mathcal{A}^+$ or $\mathcal{A}^-$ depending on the case) and two branches are created; one setting $F_{(i,j)} = 0$ and another setting $F_{(i,j)} = 1$. Should case (c) occurs, then two edges whose flows are fractional are selected, $(i,j) \in \mathcal{A}^+$ and $(u,v) \in \mathcal{A}^-$, and four branches are created with the following decisions:

1. $F_{(i,j)} = 0 \wedge F_{(u,v)} = 0$.
2. $F_{(i,j)} = 0 \wedge F_{(u,v)} = 1$.
3. $F_{(i,j)} = 1 \wedge F_{(u,v)} = 0$.
4. $F_{(i,j)} = 1 \wedge F_{(u,v)} = 1$.

$F_{(i,j)} = 0$ is enforced in the RMP by removing columns containing edge $(i,j)$, whereas in the PSP, edge $(i,j)$ is removed from all graphs to prevent columns containing it from being generated. To enforce $F_{(i,j)} = 1$, edges in sets $\delta^+(i) \setminus \{(i,j)\}$ and $\delta^-(j) \setminus \{(i,j)\}$ are removed from all graphs in the PSP and columns containing the edges are correspondingly removed from the RMP.

In practice, cuts (3.31), (3.32), and (3.34) are introduced into the RMP (one for every rider $i \in \mathcal{C}$ in the case of (3.34)) from the very beginning with their right-hand sides initially set to $\geq 0$. The right-hand sides are then correspondingly updated to those shown in (3.31), (3.32), and (3.34) as the algorithm progresses. Let $\mu$, $\nu$, and $\phi_i$ denote the duals

of cuts (3.31), (3.32), and (3.34) respectively. These duals are incorporated into the PSP by updating the costs of edges $(i,j) \in \mathcal{A}_d^+$ defined earlier in (3.11) to:

$$c_{(i,j)} = \begin{cases} \bar{c}(1-\nu) + \varsigma_{(i,j)}(1-\nu) - \pi_i^+ - \sigma_d - \mu - \phi_i & \forall (i,j) \in \delta^+(d) \\ \varsigma_{(i,j)}(1-\nu) - \pi_i^+ & \forall i \in \mathcal{O}^+ \setminus \{d\},\ \forall (i,j) \in \delta^+(i) \\ \varsigma_{(i,j)}(1-\nu) & \forall i \in \mathcal{D}^+,\ \forall (i,j) \in \delta^+(i) \end{cases} \tag{3.36}$$

and those of $(i,j) \in \mathcal{A}_d^-$ defined in (3.12) to:

$$c_{(i,j)} = \begin{cases} \bar{c}(1-\nu) + \varsigma_{(i,j)}(1-\nu) - \pi_i^- + \sigma_d - \mu & \forall (i,j) \in \delta^+(d) \\ \varsigma_{(i,j)}(1-\nu) - \pi_i^- & \forall i \in \mathcal{O}^- \setminus \{d\},\ \forall (i,j) \in \delta^+(i) \\ \varsigma_{(i,j)}(1-\nu) & \forall i \in \mathcal{D}^-,\ \forall (i,j) \in \delta^+(i) \end{cases} \tag{3.37}$$

### 3.4.5 Implementation Strategies

Several strategies are adopted in our implementation to reduce its execution time. Firstly, since the PSP involves solving at most $2n$ ESPPRCs which are independent, they are solved in parallel and multiple columns are added to the RMP in each column-generation iteration.

Secondly, to check for the convergence of the column-generation phase, a primal upper bound and a dual lower bound are maintained for the optimal objective value, $z^*$. The objective value of the RMP after each iteration, $z_{\text{RMP}}$, serves as the primal upper bound while the lower bound proposed by Lübbecke and Desrosiers (2005) is used as the dual lower bound. It is given by $z_{\text{LB}} = z_{\text{RMP}} + rc^*\lambda$, where $rc^*$ is the smallest route reduced cost discovered in the PSP and $\lambda$ is an upper bound to the number of selected routes, i.e., $\lambda \geq \sum_{r \in \Omega^+ \cup \Omega^-} X_r$. In this case, it is easy to see that $\lambda$ can be chosen as $2n$.

Let $\chi_{\text{RMP}}$ and $\chi_{\text{LB}}$ be the upper and lower bounds to the total number of selected routes, obtained by considering only the fixed cost contributions to $z_{\text{RMP}}$ and $z_{\text{LB}}$ respectively. Since the number of selected routes must be even for an integer solution, the column generation is first suspended when $2\lceil \chi_{\text{RMP}}/2 \rceil - \chi_{\text{LB}} < 2$. Cut (3.31) is then introduced to the RMP to round up the total to the nearest even integer after which the column generation is resumed. Since the optimal objective value of the MP must be integral, the column generation is terminated when $\lceil z_{\text{RMP}} \rceil - z_{\text{LB}} < 1$, after which cut (3.32) is introduced.

Finally, the branch-and-bound tree is explored depth first to quickly obtain integer solutions. During the tree exploration, a best integer solution may be obtained at any tree node by solving the RMP as a MIP (in practice, this is only done for every 1,000 tree nodes

explored beginning with the root node due to its potentially high expense). Let $z_{\text{MIP}}$ denote the objective value of the MIP solution, $z^*_{\text{int}}$ be that of the optimal integer solution sought, and $z^*_{\text{min}}$ be the smallest $z^*$ from all unexplored tree nodes. Since at any stage of the tree exploration, $z^*_{\text{min}} \leq z^*_{\text{int}} \leq z_{\text{MIP}}$, it is terminated when $z_{\text{MIP}} - z^*_{\text{min}} < 1$, at which point the optimal integer solution is given by the best integer solution.

### 3.4.6 The Root-Node Heuristic

To assess the algorithm's ability to produce high-quality solutions in an operational setting, a heuristic is conceived based on the BPA. It simply executes column generation at the root node of the branch-and-price tree within an allocated time budget $t_{\text{RMP}}$, and then finds an integer solution by solving the RMP as a MIP within another time budget $t_{\text{MIP}}$. The lexicographic objective function is simplified to only minimize the number of selected routes by setting route costs $\mathbf{c}_r \equiv \mathbf{1}$. The quality of the heuristic solution is assessed by calculating its optimality gap, given by $(z_{\text{MIP}} - z_{\text{LB}})/z_{\text{MIP}}$, where $z_{\text{MIP}}$ is the objective value of the MIP solution and $z_{\text{LB}}$ is its lower bound. Bound $z_{\text{LB}}$ is given by the optimal objective value of the RMP at convergence, $z^*$. Should the RMP not converge within $t_{\text{RMP}}$, a dual lower bound to $z^*$ that is calculated using the method proposed by Farley (1990) is used instead. Farley's lower bound is given by:

$$z_{\text{LB}} = z_{\text{RMP}} \frac{c_{r'}}{\boldsymbol{\pi}^\intercal \boldsymbol{a}_{r'}} \tag{3.38}$$

where $r' = \arg\min_{r \in \Omega^+ \cup \Omega^-} \{c_r / \boldsymbol{\pi}^\intercal \boldsymbol{a}_r : \boldsymbol{\pi}^\intercal \boldsymbol{a}_r > 0\}$, $\boldsymbol{\pi}$ is the dual optimal solution of the RMP, and $\boldsymbol{a}_r$ is the column of constraint coefficients of route $r$. The unit route costs simplify the lower bound to $z_{\text{LB}} = z_{\text{RMP}}/(1 - rc^*)$.

An alternate variant of the heuristic which relaxes forbidden paths in the RCSPA is also considered. The consideration is made based on a couple of preliminary observations: (1) Preliminary solutions to the RCSPA are very rarely infeasible, and (2) forbidding discovery of infeasible paths in the RCSPA is expensive. A consequence of this relaxation is that infeasible routes may be introduced into the RMP and therefore: (1) They will need to be filtered out before the RMP is solved as a MIP, and (2) the RMP may converge to a weaker lower bound, $z_{\text{LB}} \leq z^*$. Despite the potential loss in solution quality, the relaxation strategy may still be worthwhile as the loss may be very small and it may be outweighed by the gains resulting from shorter computation times.

## 3.5 The Clustering Algorithms

Similar to Chapter 2, this work first clusters the commuters into artificial neighborhoods with no more than $N$ commuters before the CTSP is solved for each cluster independently. It is the mechanism by which problem instances of different sizes are constructed for the following computational evaluations. The clustering strategy is adopted for the same reasons described in Section 2.2.1: It is necessary to produce computationally tractable problem instances, as preliminary evaluations have revealed that a global solution cannot be obtained for the dataset considered within a time frame that is reasonable for an operational setting, and it is also a decomposition technique that is in line with recommendations from other works, e.g. Agatz et al. (2012). The same caveats mentioned in Section 2.2.1 therefore also apply here: While the technique decomposes the dataset into smaller, independent subproblems which can then be solved concurrently, the tractability gained comes at the price of the preclusion of a global optimal solution. Nevertheless, as the following computational results will show, an attempt is still made to estimate a global optimal solution using the root-node heuristic, albeit with an inordinate time budget.

This work considers two clustering techniques. The primary technique, referred to as the *spatial* clustering algorithm, is identical to that described in Section 2.2.1. As a brief recap, it groups commuters by the spatial proximity of their home locations by first treating each as a point in $\mathbb{R}^2$. A point for commuter $c$ is simply a 2-dimensional position vector, $\boldsymbol{x}_c^\intercal = \{x_c, y_c\}$, whose components $x_c$ and $y_c$ are given by the x- and y-coordinates of the commuter's home on the Cartesian plane. The algorithm itself proceeds in a similar manner to the $k$-means clustering algorithm (Lloyd 1982) with $k = \lceil |\mathcal{C}|/N \rceil$, with a small exception to its assignment step which assigns at most $N$ commuters to each cluster. The assignment is accomplished by solving a generalized-assignment MIP whose objective is to minimize the total Euclidean distance between every point and their assigned cluster centers.

The secondary technique considered is motivated by the desire to evaluate the efficacy of grouping commuters based on the temporal proximity of their trips in addition to the spatial proximity of their homes. To accomplish this goal, each commuter $c$ is represented by a 4-dimensional position vector, $\boldsymbol{x}_c^\intercal = \{x_c, y_c, dt_c^+, at_c^-\}$. The first two components, $x_c$ and $y_c$, are identical to those used in the spatial clustering technique; they are the spatial coordinates of the commuter's home. As every commuter makes two trips per day, the other two components account for the times of these trips; $dt_c^+$ denotes the time the commuter departs from her home for her inbound trip, while $at_c^-$ denotes the time the commuter arrives at her home for her outbound trip. This algorithm is otherwise similar to the spatial

clustering algorithm; its only difference is that it represents the commuters as points in $\mathbb{R}^4$ and utilizes the Euclidean distances between those points. It is therefore still an algorithm that groups no more than $N$ commuters into clusters, albeit based on the *spatio-temporal* proximity of the origins of their inbound trips and the destinations of their outbound trips. As the position vectors in this algorithm contain components with vastly different units and scales, the values from each dimension are first standardized before the algorithm is executed; they are scaled so that the values from each dimension have zero mean and unit variance. This is done so that each dimension has an approximately equal contribution to the Euclidean distances used in the algorithm. In other words, the standardization is performed to prevent the distance calculations from being biased towards or dominated by the components of any one dimension. This second technique is referred to as the *spatio-temporal* clustering algorithm for obvious reasons.

## 3.6 Experimental Results

This section reports the computational results for the proposed algorithms, as well as their effectiveness in reducing parking pressure.

### 3.6.1 Experimental Setting

The computational performance of the algorithms is evaluated using problem instances derived from the Ann Arbor commute-trip dataset. The experiments in this chapter focus on the trips made by the approximately 3,900 commuters living within Ann Arbor's city limits (the region bounded by highways US-23, M-14, and I-94), an area spanning 27 square miles, from which approximately 2,200 commute trips are made on a daily basis.

Several assumptions are made regarding commuters using the trip-sharing platform. Firstly, it is assumed that when requesting a commute trip, rider $i$ would specify the desired arrival time at the destination of her inbound trip, $at_i^+$, and the desired departure time at the origin of her outbound trip, $dt_i^-$. This assumption is consistent with that made in other DARP literature, e.g. Jaw et al. (1986), Cordeau and Laporte (2003b), and Cordeau (2006). It is also assumed that the commuters are willing to tolerate a maximum shift of $\pm\Delta$ to the desired times. Therefore, by treating the arrival and departure times to and from the parking structures as the desired times, time windows of $[a_{n+i}, b_{n+i}] = [at_i^+ - \Delta, at_i^+ + \Delta]$ and $[a_i, b_i] = [dt_i^- - \Delta, dt_i^- + \Delta]$ are associated with the destination of the inbound trip and the origin of the outbound trip of rider $i$ respectively. Consequently, time windows at the origin of the inbound trip and at the destination of the outbound trip of rider $i$ are calculated using

$[a_i, b_i] = [a_{n+i} - \zeta_i - L_i, b_{n+i} - \zeta_i - \tau_{(i,n+i)}]$ and $[a_{n+i}, b_{n+i}] = [a_i + \zeta_i + \tau_{(i,n+i)}, b_i + \zeta_i + L_i]$ respectively. It is also assumed that each commuter is willing to tolerate at most an $R\%$ extension to her direct-ride duration, i.e., $L_i = (1+R)\tau_{(i,n+i)}$. This assumption is similar to that made by Hunsaker and Savelsbergh (2002). Finally, for the spatio-temporal clustering algorithm, the departure and arrival times of the inbound and outbound trips of rider $i$ are defined as follows: $dt_i^+ = at_{n+i}^+ - \zeta_i - \tau_{(i,n+i)}$ and $at_i^- = dt_{i-n}^- + \zeta_{i-n} + \tau_{(i-n,i)}$.

### 3.6.2 Algorithmic Settings

The clustering algorithms are used to construct problem instances of different sizes by varying $N$. Due to the non-deterministic nature of their initialization step, the algorithms are executed 100 times for each value of $N$, after which only the solution with the smallest assignment objective value is selected. The shortest path, travel-time estimate, and travel-distance estimate between any two locations are obtained using the GraphHopper Directions API which uses data from OpenStreetMap. All algorithms are implemented in C++ with parallelization duties being handled by OpenMP. The resource-constrained shortest path function from Boost 1.64.0's Graph Library is used to implement the RCSPA, while Gurobi 7.5.1 is invoked to solve all LPs and MIPs. The route fixed cost $\bar{c}$ is obtained by making a very conservative overestimate of the longest route length. The RMP of the BPA is initialized with the set of all feasible single- and two-trip routes, which is generated using the REA with $K = 2$. Each problem is solved on a high-performance computing cluster using 12 cores of a 2.5 GHz Intel Xeon E5-2680v3 processor and 64 GB of RAM. Unless stated otherwise, a time limit of 12 hours is applied to all problems and the best feasible solution is reported for those that cannot be solved optimally within the time limit.

### 3.6.3 Selecting Values for $\Delta$ and $R$

Half of the time-window size, $\Delta$, and the ride-duration limit, $L_i = (1 + R)\tau_{(i,n+i)}$, directly influence the QoS of rider $i$; the former represents the maximum amount of time by which the rider needs to shift (up or down) her desired arrival time to or departure time from a parking lot, whereas the latter represents the maximum amount of time the rider has to spend on the vehicle. Therefore, it is ideal for any rider to have the values of $\Delta$ and $R$ be as small as possible. However, doing so will also limit the potential for trip sharing. Indeed, selecting values for either parameter involves a trade-off between user convenience and trip shareability. A sensitivity analysis was therefore performed to study the impact of these two parameters on the vehicle reduction of the CTSP algorithm, by first applying the spatial clustering algorithm with $N = 100$ on the commuters traveling on each of the first

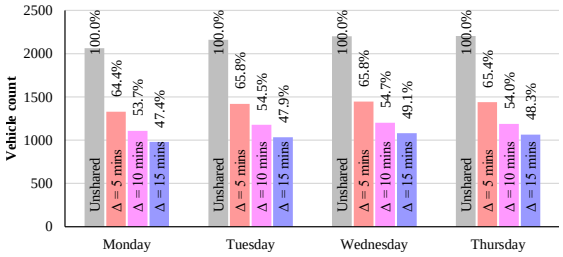



Figure 3.3: Effect of Increasing $\Delta$ on Total Vehicle Count ($R = 50\%$).

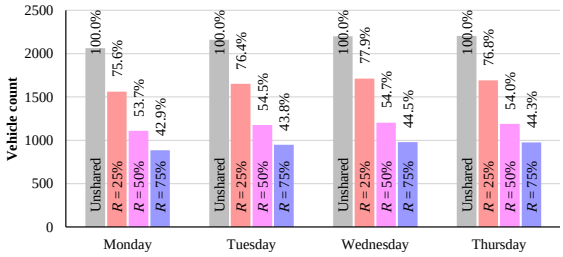



Figure 3.4: Effect of Increasing $R$ on Total Vehicle Count ($\Delta = 10$ mins).

four weekdays of week 2, and then optimizing trip sharing within each cluster using the REA with $K = 4$ (the capacity of a car). The following values were used in the analysis: $\Delta \in \{5, 10, 15\}$ mins and $R \in \{25, 50, 75\}\%$.

Figures 3.3 and 3.4 summarize the results of the analysis for $\Delta$ and $R$ respectively. They include the required number of vehicles under the existing no-sharing conditions as well as the percentage of each vehicle count as a fraction of the no-sharing count for additional perspective. The figures quantify the trade-off mentioned earlier; while the smaller values of $\Delta = 5$ mins and $R = 25\%$ may be convenient for the riders, the results indicate that the vehicle-reduction potential is significantly hampered by these values. Vehicle reduction increases, albeit at a decreasing rate, as both $\Delta$ and $R$ are increased. While the largest values of both parameters produce the best vehicle-reduction potential, they also demand the highest amount of tolerance to inconvenience from the riders. Therefore, $\Delta = 10$ mins and $R = 50\%$ were deemed to be the best compromise, as they still produced a sizeable amount of vehicle reduction while not inducing a significant amount of inconvenience to the riders. These values are therefore used in the rest of the experiments.

### 3.6.4 Vehicle Capacity Scaling

The next set of computational experiments explores the scalability of the REA and the BPA with increasing vehicle capacity. A variety of car-pooling programs provide small vans to commuters: These vans can typically carry up to 8 people and it is important to evaluate the benefits of using such vehicles. Problem instances are created by applying the spatial clustering algorithm with $N \in \{75, 100\}$ on commuters traveling on a selected day (Wednesday of week 2, which had 2,200 commute trips) and setting $K \in \{4, 5, 6, 7, 8\}$. Let $n$ denote the size of a cluster. Since $N$ only controls the upper bound for the size of clusters produced by the algorithm, residual clusters with $n < N$ are also generated when the total number of commuters, $|\mathcal{C}|$, is not an exact multiple of $N$. For the experiments, only clusters with sizes of exactly 75 and 100 are selected as the main problem instances (24

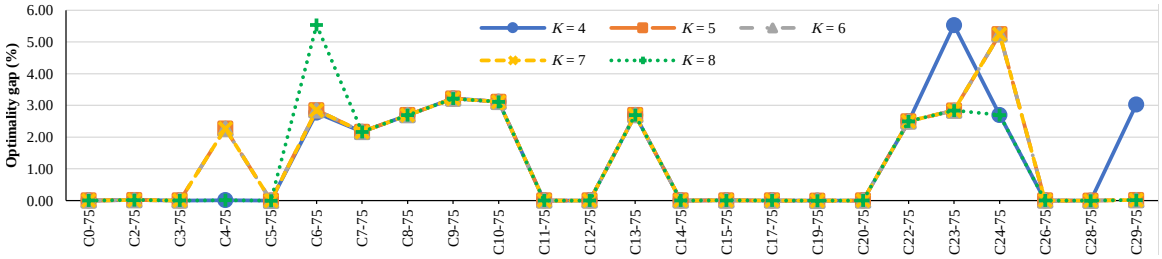



Figure 3.5: Optimality Gap of MIP Solution at Root Node of BPA for Problem Instances with $n = 75$.

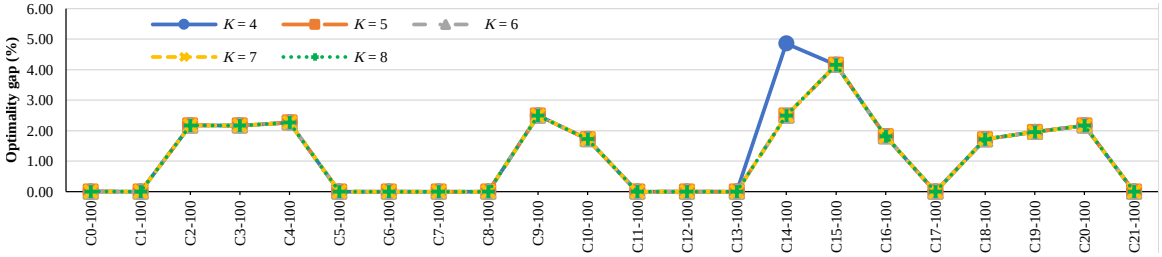



Figure 3.6: Optimality Gap of MIP Solution at Root Node of BPA for Problem Instances with $n = 100$.

clusters with $n = 75$ and 22 with $n = 100$) for in-depth study. Detailed results of the REA and the BPA on these selected clusters are presented in Tables A.1 and A.2 respectively in Appendix A. However, Figures 3.13–3.15 which aggregate vehicle count, route distance, and average ride duration for the day *utilize results from all clusters*.

*The REA is unable to complete route enumeration within the time limit when $K > 6$, therefore results of the algorithm for $K \in \{7, 8\}$ are not available*. This is not surprising, owing to the $\mathcal{O}(n^K)$ time complexity of its route-enumeration phase. The time limit for clusters C9-100 and C20-100 when $K = 6$ also had to be extended to obtain a solution. While the BPA is able to handle larger vehicle capacities for the most part, it could not find a root-node solution within the time limit for cluster C10-75 when $K = 8$, so the time limit for this case had to be extended too. As expected, when problems are solved to optimality, identical objective values are produced by both algorithms as shown by the same vehicle counts and total route distances in their results.

The REA produces optimal results in all instances when $K \leq 6$, while the BPA does so for all but 14 instances. Unsurprisingly, these 14 instances are typically characterized by large vehicle capacities ($K \geq 5$) as well as relatively large edge counts. *For these instances, the optimality gap of the best feasible solution is consistently $< 5\%$, and a comparison of their vehicle-count results against those of the REA that are available reveals that they are in fact optimal*. Also notable is the number of columns generated by the BPA being

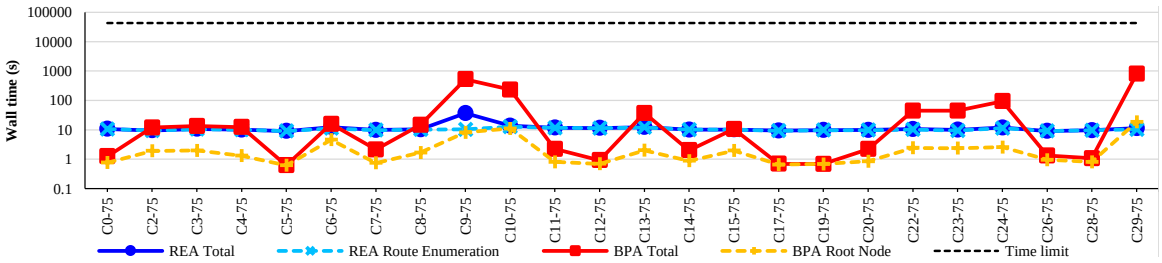



Figure 3.7: Computation Times for Problem Instances with $n = 75$ and $K = 4$.

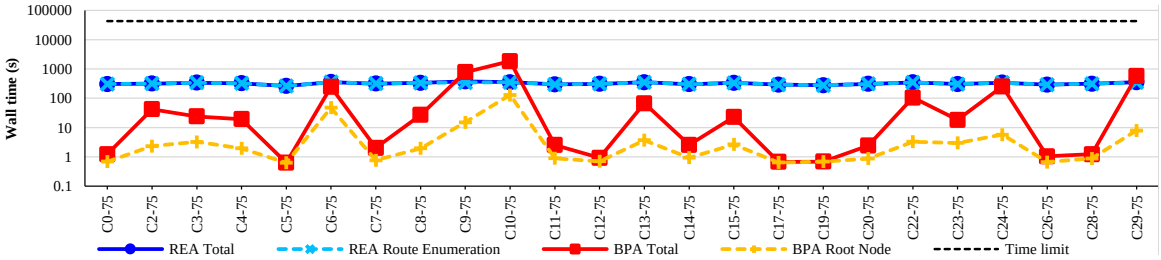



Figure 3.8: Computation Times for Problem Instances with $n = 75$ and $K = 5$.

consistently less than the REA, and in some cases significantly so. Another notable result is the excellent quality of the BPA's root-node solution which is summarized in Figures 3.5 and 3.6 for problem instances with $n = 75$ and $n = 100$ respectively. Its optimality gap is $< 6\%$ in all instances and is zero for some cases, signaling that it is a viable option when optimality is not crucial. The integrality gap, also being $< 6\%$ for all instances, emphasizes the strength of the primal lower bound provided by the RMP's optimal objective value.

Lastly, another notable observation is the disparity in the total number of feasible inbound and outbound edges of the graphs of the BPA. Recall that feasible edges are those that satisfy the a priori feasibility constraints outlined in Section 3.4.2. The edge counts can be seen as a rough indicator of the shareability potential of the set of trips being considered, and the number of outbound edges being less than inbound edges in all problem instances indicates fewer sharing opportunities for outbound trips. This can be attributed to the wider distribution of their departure times as was shown in Figure 1.2 which further complicates ride sharing. It also highlights another unique challenge to solving the CTSP, as maximal sharing is sought over two sets of trips (inbound and outbound) with different shareability potential.

Figures 3.7–3.9 summarize the computation times of both algorithms for the problem instances with $n = 75$ and $K \in \{4, 5, 6\}$, while Figures 3.10–3.12 do the same for $n = 100$ and $K \in \{4, 5, 6\}$. The figures reveal that the computation times of the REA are more consistent across problem instances with the same $n$ and $K$ values. The times also appear to be dominated by the route-enumeration phase for these instances. The figures also show

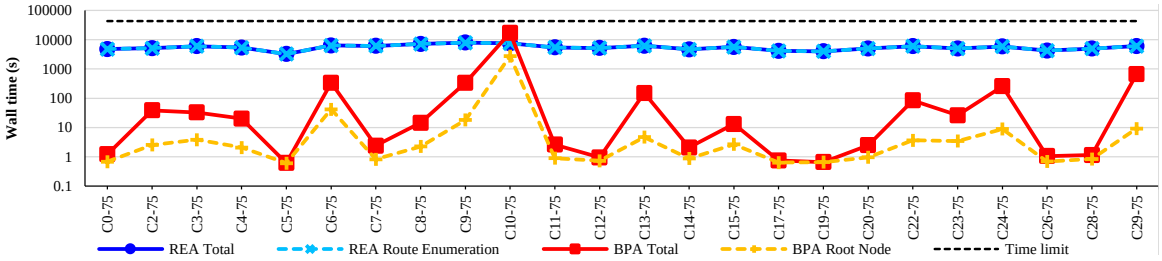



Figure 3.9: Computation Times for Problem Instances with $n = 75$ and $K = 6$.

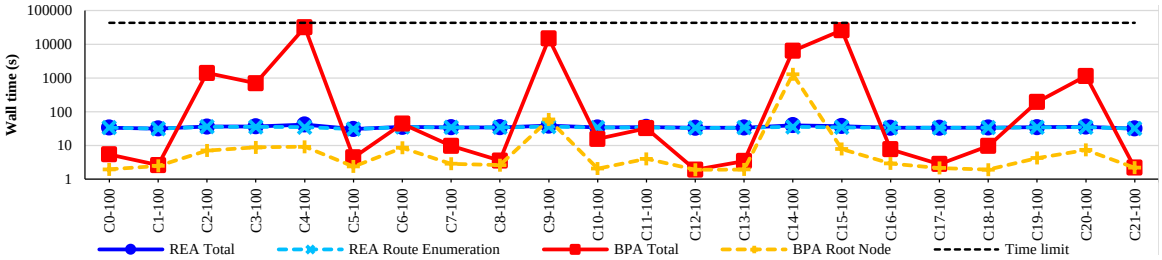



Figure 3.10: Computation Times for Problem Instances with $n = 100$ and $K = 4$.

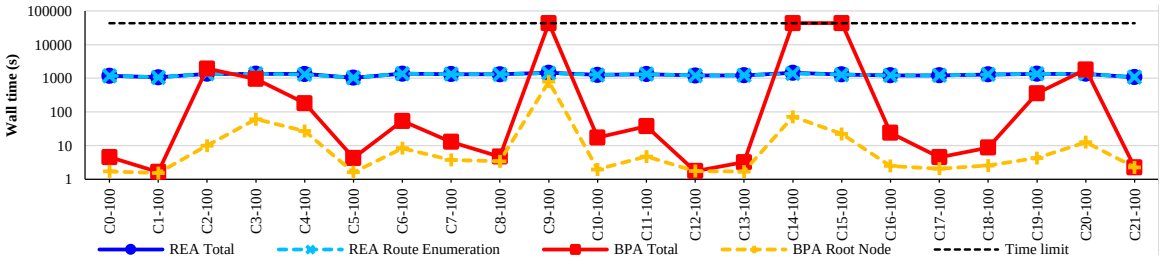



Figure 3.11: Computation Times for Problem Instances with $n = 100$ and $K = 5$.

that computation times of the REA are more sensitive to $K$; they appear to increase more rapidly with increasing $K$ than the BPA. The former's sensitivity can be attributed to its route-enumeration phase's $\mathcal{O}(n^K)$ time complexity and the fact that it is dominating the total computation times in these instances. The BPA is slower in 13 out of 24 instances when $K = 4$ and $n = 75$, and it is slower in nine out of 22 instances when $K = 4$ and $n = 100$. These fractions decrease however as $K$ becomes larger to the point where the BPA is faster in all but one instance when $K = 6$ and $n = 75$ and in all but three instances when $K = 6$ and $n = 100$. These observations, combined with the results showing the BPA's ability to obtain solutions for $K > 6$, indicate that the BPA scales better with increasing vehicle capacity. Also noteworthy is the time taken to produce the root-node solution for the BPA; it is faster than the REA in all but one instance when $n = 75$, and in all but two instances when $n = 100$. This further strengthens the case for it being a viable option when an optimal solution is not sought.

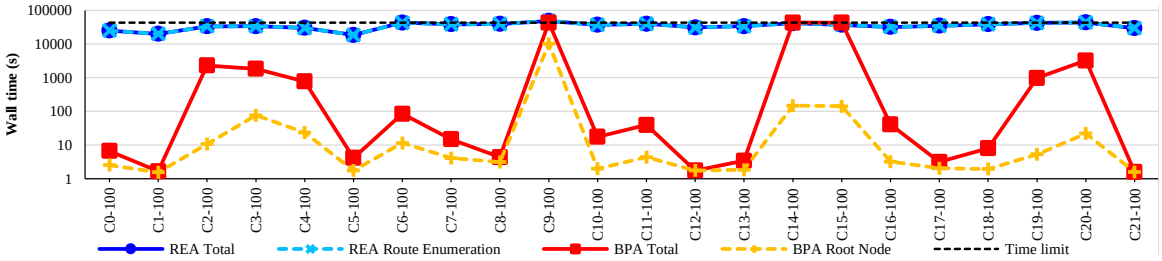



Figure 3.12: Computation Times for Problem Instances with $n = 100$ and $K = 6$.

Finally, Figures 3.13, 3.14, and 3.15 summarize the effect of increasing $K$ on total vehicle count, total distance of selected routes, and average ride time per commuter respectively. They show aggregated results from all clusters from a single weekday (Wednesday of week 2) as $N$ is kept constant. The percentage of each quantity as a fraction of its value when $K = 1$ as well as the results for $K \in \{1, 2, 3\}$ are included for additional perspective. Firstly, Figures 3.13 and 3.14 reveal diminishing marginal decreases in total vehicle count and total travel distance as $K$ is increased. Furthermore, the benefit of increasing vehicle capacity almost diminishes completely beyond $K = 3$. This can be attributed to the nature of the routes in the CTSP being very short. Each needs to start and end at the origin and destination of its driver respectively, and ride-duration limits are imposed on the driver in addition to all passengers. The length of each route is therefore constrained by the ride-duration limit of its driver. As longer routes are needed to fully utilize the capacities of larger vehicles (to pick up and drop off more riders), *the routes of this problem do not benefit from larger vehicle capacities*. For this reason, subsequent computational experiments only consider the use of cars by limiting $K$ to 4. Figure 3.15 provides a first glimpse of the trade-off to ride sharing: Increased average ride durations. As vehicle capacity is increased, so does ride-sharing opportunities. Consequently, an increase in ride duration should also be expected as a ride is shared with more and more people. There appears to be an inverse relationship between average ride duration and vehicle count or total travel distance, as the former increases as either of the latter decreases. A similar diminishing effect in the marginal increase of average ride duration is also seen as vehicle capacity is increased.

### 3.6.5 Cluster Size Scaling

The next set of experiments explores the scalability of the REA and the BPA with increasing cluster size. To this end, the spatial clustering algorithm is applied on the commuters traveling on each of the first four weekdays of week 2 (which had 2065, 2161, 2200, and 2203 commute trips respectively) with $N \in \{200, 300, 400\}$ (results for $N \in \{75, 100\}$ are

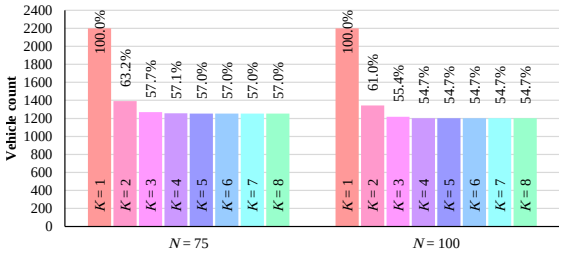



Figure 3.13: Effect of Increasing Vehicle Capacity on Total Vehicle Count.

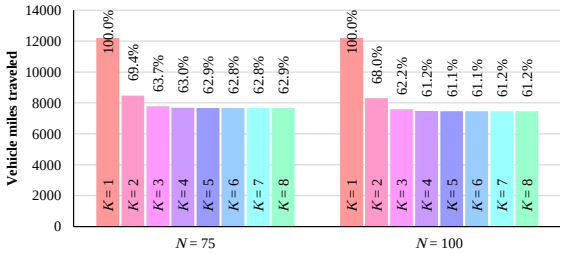



Figure 3.14: Effect of Increasing Vehicle Capacity on Total Route Distance.

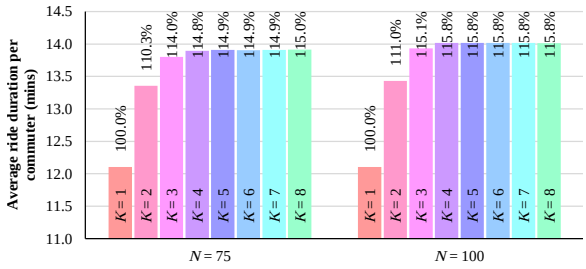



Figure 3.15: Effect of Increasing Vehicle Capacity on Average Ride Duration.

already available from Section 3.6.4). $K$ is fixed to 4 in all experiments. Detailed results of the REA and the BPA, shown in Tables A.3 and A.4 respectively in Appendix A, and the rest of the discussions in this section focus on selected clusters with sizes of exactly 200, 300, and 400 (11 with $n = 200$, 13 with $n = 300$, and 8 with $n = 400$) as the results are intended to show the effect of progressively increasing the cluster size by increments of 100. However, results from all clusters, including residuals with $n < N$, are used in Figures 3.21, 3.22, and 3.23 which show aggregate vehicle count, route distance, and average ride duration respectively for each day.

Figure 3.16 summarizes the number of instances that can be solved optimally by both algorithms together with the total number of instances considered for each $n$ value and the percentage of each quantity as a fraction of the total. When $n = 200$, the REA is able to obtain the optimal solution for all but one instance. Conversely, the BPA is only able to produce optimality for four instances. As $n$ is increased, so do the size and complexity of the problem instances as evident from the number of columns generated and the edge counts, which lead to fewer instances being solved to optimality by either algorithm. When $n = 400$, none of the instances could be solved to optimality by the BPA, and only five out of eight instances could be solved optimally by the REA.

Figure 3.17 displays the optimality gaps produced by both algorithms in these experiments. The optimality gaps of suboptimal solutions of the REA are excellent, being $< 0.5\%$ for all instances. Moreover, the optimal vehicle count is obtained in all these instances. The

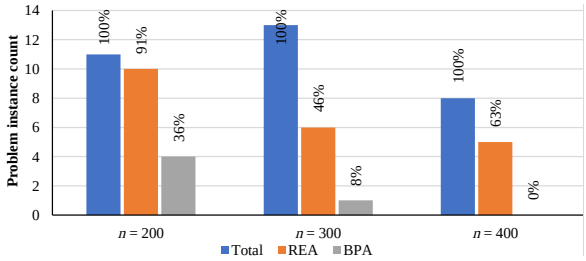



Figure 3.16: Number of Problem Instances Solved Optimally when $n \in \{200, 300, 400\}$.

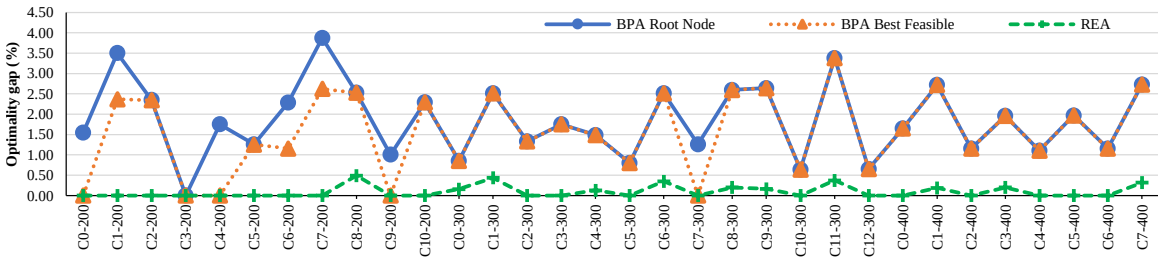



Figure 3.17: Optimality Gaps of the REA and the BPA for Problem Instances with $n \in \{200, 300, 400\}$.

optimality gaps of suboptimal BPA solutions are competitive, being $< 4\%$ in all instances, however there are a few instances where the optimal vehicle count is not obtained, e.g. clusters C8-200, C3-300, and C0-400. For these instances, the vehicle count is typically off by one when compared to the optimal counts of the REA. Root-node solutions of the BPA remain excellent as their optimality gaps are also $< 4\%$ in all of the instances considered, with it being optimal for one instance (cluster C3-200). The total number of columns generated by the BPA also remain fewer than the REA in all instances.

Figures 3.18–3.20 compare the computation times of both algorithms for all problem instances with $n \in \{200, 300, 400\}$. When combined with Figures 3.7 and 3.10, they provide a clear picture of how both algorithms scale with increasing cluster size as $K$ is kept constant at 4. Figure 3.18 shows the REA being faster then the BPA in eight out of the 11 instances tested when $n = 200$. The allocated time quickly gets saturated in most problem instances by either algorithm as $n$ is increased to the the point where the BPA reaches the time limit in all problem instances when $n = 400$, while the REA also does so for three out of the eight instances tested. The computation times of the REA are still dominated by its route-enumeration phase in most instances, however there also exist a few instances where solving the MP consumes a bigger portion of the total computation times, e.g. cluster C8-200 in Figure 3.18. This is due to the increased complexity of solving the MIP from the higher column counts. More fluctuations are also observed in the computation times of the REA across problems with the same $n$ value, and this too can be

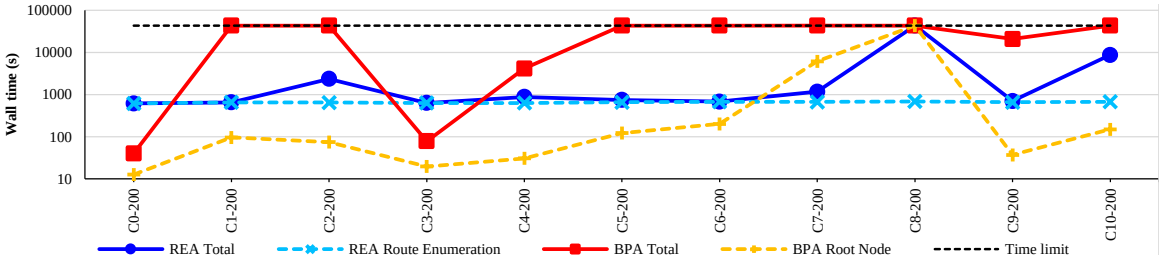



Figure 3.18: Computation Times for Problem Instances with $n = 200$.

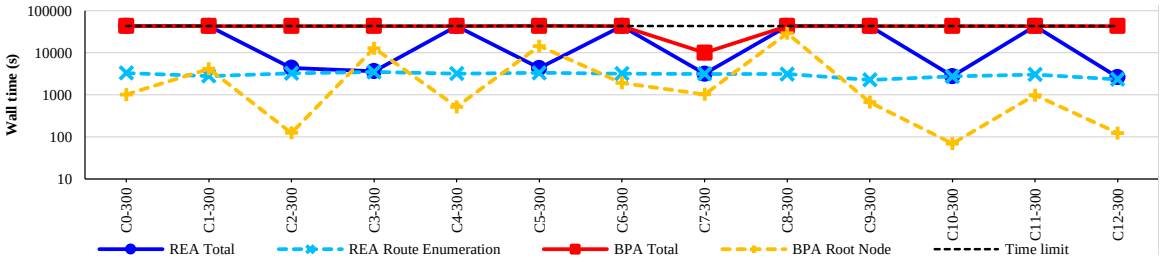



Figure 3.19: Computation Times for Problem Instances with $n = 300$.

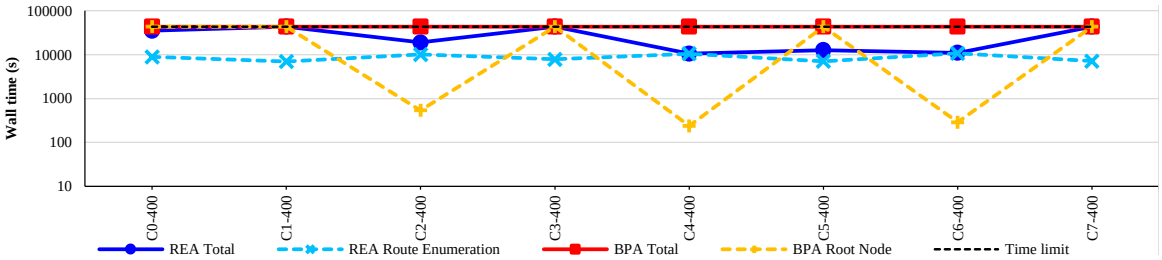



Figure 3.20: Computation Times for Problem Instances with $n = 400$.

attributed to the complexity of solving the MIP as the time for the route-enumeration phase remains consistent for all instances. The root-node solution of the BPA remains a viable option if a quick, high-quality solution is required, as it is faster than the REA in all but nine instances. The results indicate that the REA is a better choice for large cluster sizes if an optimal solution, or one that is as close to optimal as possible, is sought. However, the root-node solution of the BPA is the best option if only a high-quality solution is needed, as it is faster than the REA in most of the instances tested.

Finally, Figures 3.21, 3.22, and 3.23 illustrate the effect of increasing $N$ on the overall results in terms of the total vehicle count, the total distance of selected routes, and the average ride time per commuter. The figures show aggregated results from all clusters for the first four weekdays of week 2. Results for unshared trips as well as the value of each quantity as a percentage of their corresponding unshared values are included for added context. Figure 3.21 also includes the estimated vehicle count results of the CTSP when

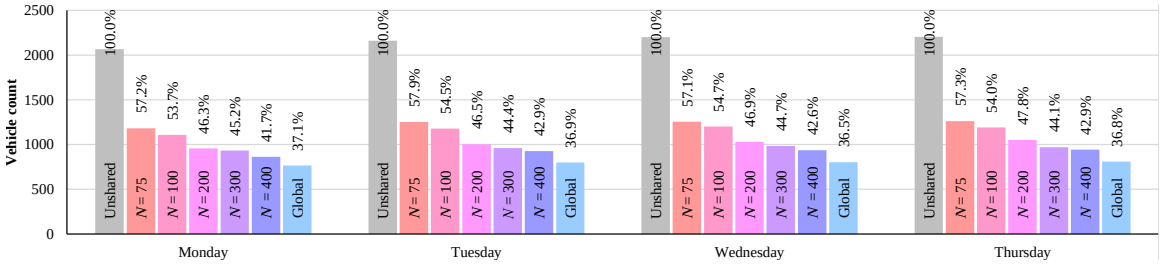



Figure 3.21: Effect of Increasing Cluster Size on Total Vehicle Count.

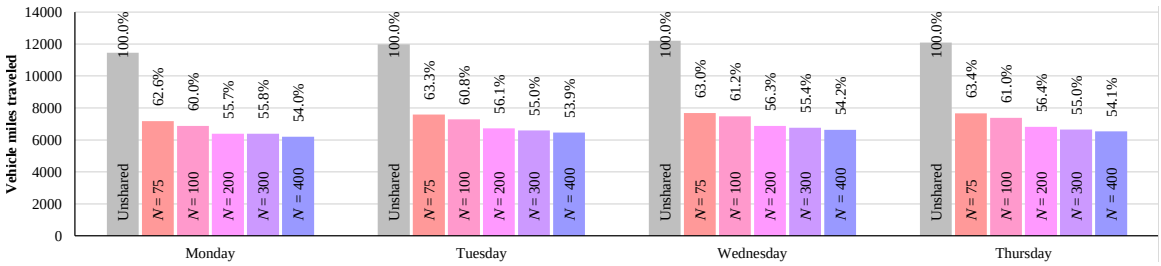



Figure 3.22: Effect of Increasing Cluster Size on Total Route Distance.

the commuters are not clustered (labeled "Global" in the figure) for even more perspective. These global results are obtained using the root-node heuristic (the variant that relaxes forbidden paths) with a massive time budget of $t_{\text{RMP}} = 24$ hours and $t_{\text{MIP}} = 24$ hours, and their optimality gaps range between 4.0–5.5%.

As expected, the total vehicle count and total route distance results improve as $N$ is increased, as larger $N$ values produce larger 'neighborhoods' which provide increased ride-sharing opportunities. Figure 3.21 also indicates that the total vehicle counts can be further reduced by approximately another 5–6% if the commuters are not clustered at all, however these results can only be obtained at the steep price of using an inordinate time budget. Average ride duration also increases with $N$ as expected. Diminishing marginal decreases (respectively increases) in total vehicle count and total route distance (respectively average ride duration) are also observed with increasing $N$, however the effect is not as pronounced as that from increasing $K$, signaling that increasing maximum cluster size is more effective at improving ride-sharing results than increasing vehicle capacity for the CTSP.

### 3.6.6 Generalizability of Vehicle Reduction Results

To assess the generalizability of the vehicle reduction results shown earlier (e.g., in Figure 3.21), additional datasets are artificially constructed by randomly permuting the trip itineraries of the population considered; the commuting destinations together with the corresponding arrival and departure times to and from the destinations of every commuter

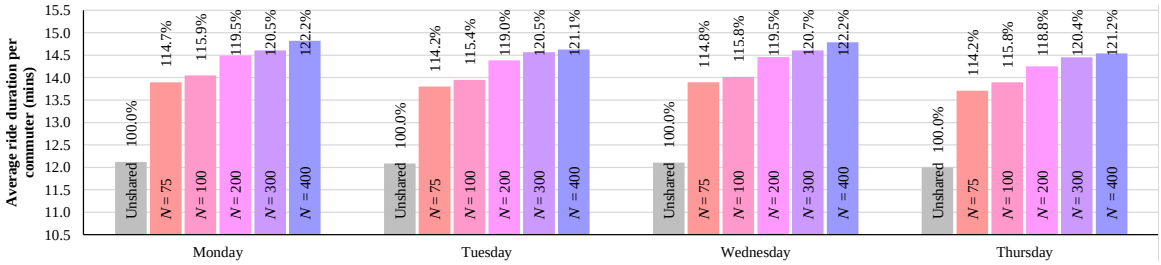



Figure 3.23: Effect of Increasing Cluster Size on Average Ride Duration.

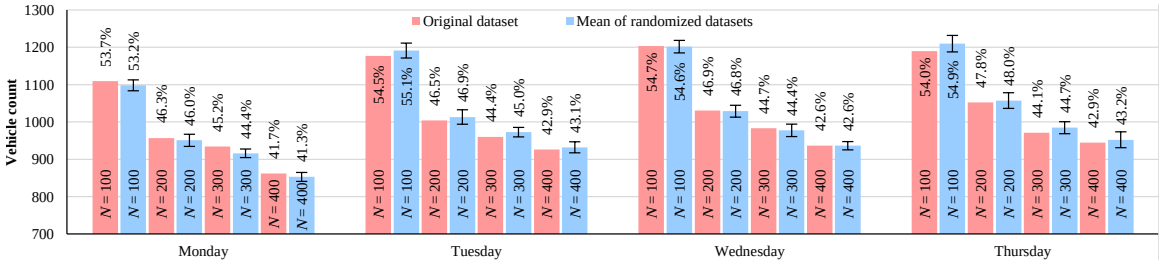



Figure 3.24: Aggregated CTSP Vehicle Counts for Randomized Datasets.

in the original dataset are randomly shuffled. This random permutation is performed to eliminate any underlying temporal structure that may exist in the original dataset. The performance of the CTSP algorithm on the randomized dataset is then evaluated by first applying the spatial clustering algorithm with $N \in \{100, 200, 300, 400\}$ and then applying the REA on every cluster to generate the CTSP routing plan. This experiment is repeated on 20 different random permutations of the dataset to produce 20 sets of results. The results are summarized in Figure 3.24. It shows the mean of the aggregated vehicle counts of the 20 random permutations for each $N$ value, with error bars illustrating the 95% confidence interval for each value, calculated by taking twice the standard deviation of the aggregated counts of the 20 permutations. For additional context, the figure includes the vehicle count results from the original dataset, and it also shows percentage values which represent each count as a fraction of the total number of vehicles used under the present no-sharing conditions.

Most evident from the figure is how small the variations in total vehicle counts are for the various permutations: At most, their percentages vary by $\pm 1\%$ from the mean. This signifies that despite the numerous trip variations, the CTSP algorithm is still able to produce consistent vehicle reductions. On top of that, the results of the original dataset are always within the 95% confidence intervals, except for the results on Monday when $N = 300$. Even then, its percentage differs from the mean of the randomized results by less than 1%. This once again alludes to the robustness of the vehicle count results of the

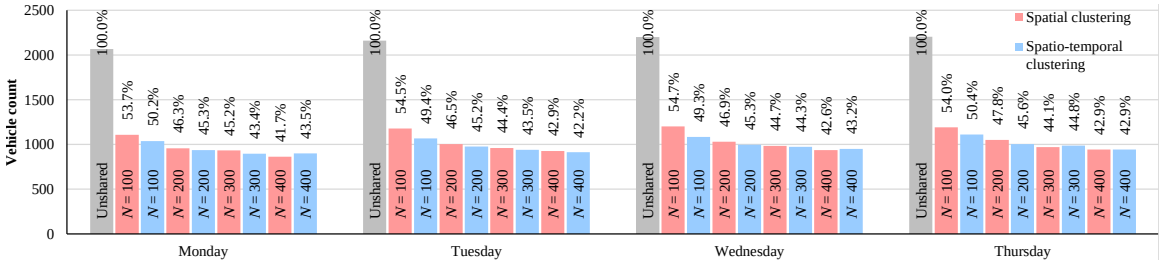



Figure 3.25: Comparison of Total Vehicle Count Results Between Spatial and Spatio-Temporal Clustering.

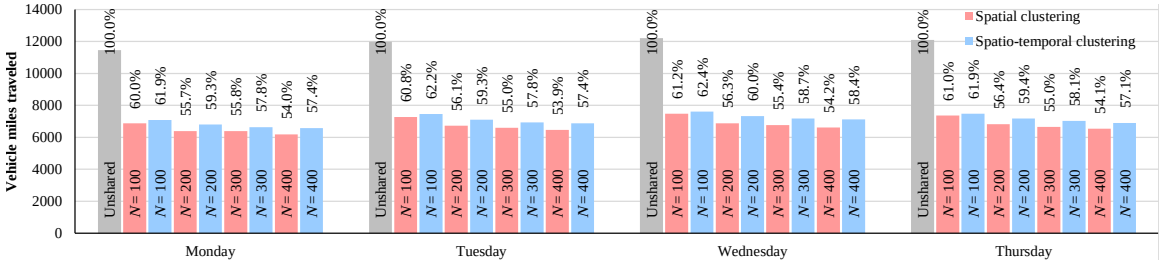


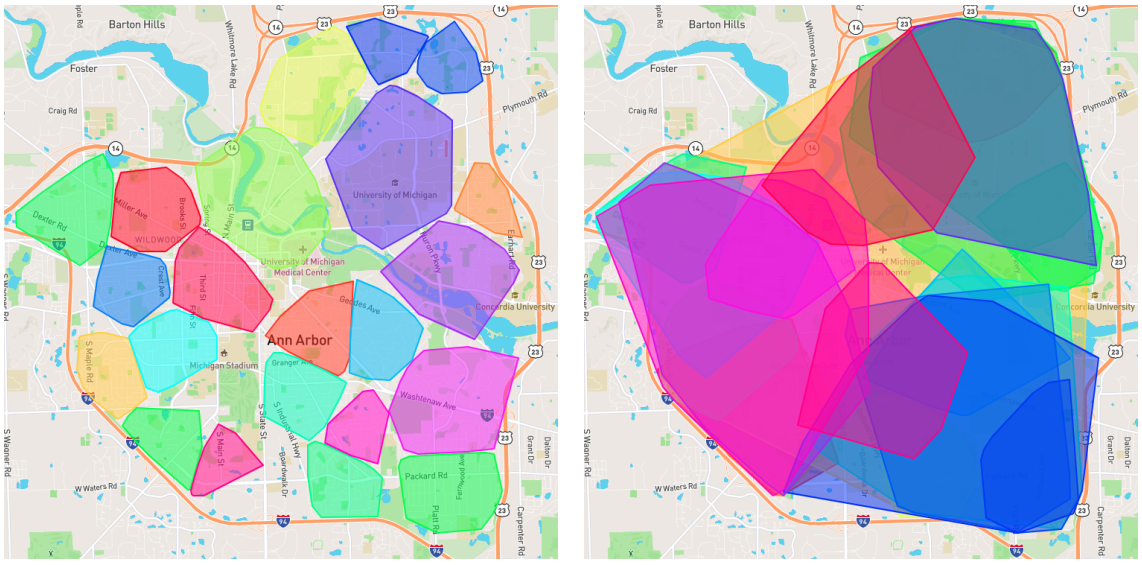

Figure 3.26: Comparison of Total Route Distance Results Between Spatial and Spatio-Temporal Clustering.

CTSP. It indicates that the magnitudes of the vehicle reductions observed earlier are not one-off results that are unique to the original dataset; similar reductions may be expected from other trip variations due to the robustness of the CTSP algorithm.

### 3.6.7 Spatial Versus Spatio-Temporal Clustering

Section 3.5 proposed a spatio-temporal clustering technique which groups the commuters based on the temporal proximity of the trip times at their commuting origins in addition to the spatial proximity of their homes. To evaluate the effectiveness of this technique, it is applied on the dataset with several $N$ values—$N \in \{100, 200, 300, 400\}$—after which the CTSP routing plan is generated for every cluster using the REA. Aggregated results from all clusters are then compared against those from the spatial clustering technique. Figure 3.25 compares the aggregated vehicle count results of both clustering techniques, while Figure 3.26 compares their aggregated route distances. Both figures include results from unshared trips and the percentages of each quantity as a fraction of the unshared values for additional context.

Firstly, the spatio-temporal clustering technique appears to be more effective at grouping commuters with similar trip itineraries for the purpose of minimizing vehicle counts when $N$ is small. Figure 3.25 shows that the spatio-temporal clusters require an average




(a) Spatial Clusters

(b) Spatio-Temporal Clusters

Figure 3.27: Visualization of Spatial and Spatio-Temporal Clusters for Wednesday of Week 2 ($N = 100$).

of 4.4% fewer vehicles than the spatial clusters when $N = 100$. This indicates that when the clusters are small, temporal proximity of the trips plays a significant role in assisting the CTSP algorithm to discover more compatible trip itineraries within each cluster, which in turn allows more trips to be combined to increase ride-sharing potential. This benefit, however, progressively diminishes as $N$ is incremented to the point where it disappears completely when $N = 400$. This indicates that the role of temporal proximity becomes less prominent as the cluster size increases, as the CTSP algorithm has increasingly larger pools of trips at its disposal which consequently makes discovery of trips with compatible itineraries less of a challenge. It also signifies that the vehicle reduction capability of the CTSP algorithm becomes more robust to the two clustering techniques considered as the problem size increases.

There is also a drawback to the spatio-temporal clustering technique, which is illustrated in Figure 3.26. Its total route distance results are consistently larger, and therefore they are comparatively poorer than those of the spatial clustering algorithm, regardless of the value of $N$. This observation can be easily explained by comparing the spatial representations of the resulting clusters. For instance, Figures 3.27(a) and 3.27(b) visualize the convex hulls of the clusters produced by each technique on the spatial plane. The spatial clustering algorithm minimizes the total Euclidean distance on the spatial plane, therefore it produces spatially separated clusters that each occupies minimal area on the plane. On the other hand, the spatio-temporal clustering algorithm minimizes the total Euclidean dis-

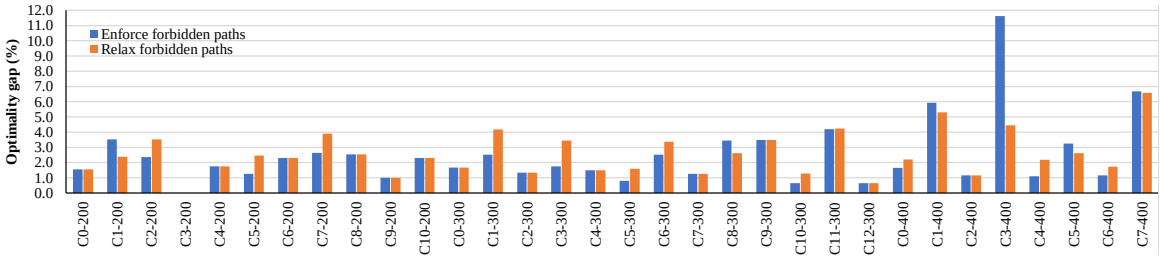



Figure 3.28: Optimality Gaps of Root-Node Heuristics for Problem Instances with $n \in \{200, 300, 400\}$.

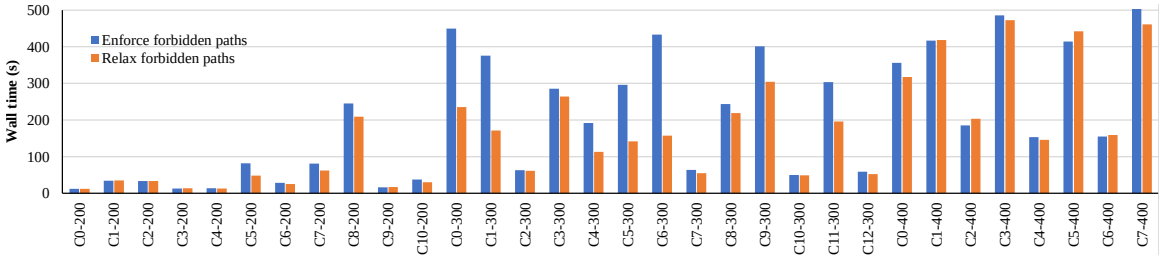



Figure 3.29: RMP Convergence Times of Root-Node Heuristics for Problem Instances with $n \in \{200, 300, 400\}$.

tance in $\mathbb{R}^4$, and Figure 3.27(b) merely shows the projection of the clusters on the spatial plane. Therefore, on the spatial plane, the spatio-temporal clusters overlap each other and they occupy relatively larger areas than the spatial clusters. As a result, the CTSP routes for the spatio-temporal clusters have to cover longer distances when picking up passengers for their inbound trips and when dropping them off for their outbound trips. This leads to the larger total distances observed in Figure 3.26.

### 3.6.8 Efficiency of the Root-Node Heuristic

Finally, the root-node heuristic is applied on the selected clusters with $n \in \{100, 200, 300, 400\}$ from Sections 3.6.4 and 3.6.5 with time budgets of $t_{\text{RMP}} = 8$ minutes and $t_{\text{MIP}} = 2$ minutes, resulting in a total time budget of 10 minutes per instance which is deemed reasonable for an operational setting. The detailed results are presented in Table A.5 in Appendix A.

Figure 3.28 compares the optimality gaps of the two variants of the heuristic—the one that enforces forbidden paths and the one that does not—for problem instances with $n \geq 200$, while Figure 3.29 compares the time spent for their RMPs to converge. It can be seen from the first figure that the optimality gaps of both variants are typically $< 5\%$, with the relaxation heuristic having an optimality gap that is only $0.1\%$ larger on average.

This minimal loss can be attributed to its small fraction of infeasible routes ($< 0.5\%$ of all generated routes in the instances tested). In some instances, the relaxation heuristic produces optimality gaps that are smaller, and this can be attributed to it being able to solve significantly more column-generation iterations within its time budget compared to the other heuristic which has to execute the expensive forbidden-path algorithm. The second figure shows the relaxation heuristic completing its column-generation phase faster in the majority of the problem instances, with it being $26\%$ faster on average than the first heuristic. Nevertheless, regardless of the variant used, the results further reinforce initial claims that the root-node heuristic is indeed able to generate provably high-quality solutions in time-constrained scenarios for problem instances of various sizes.

## 3.7 Conclusion

To relieve parking pressure which has been steadily increasing in cities and in university and corporate campuses, this chapter explores a car-pooling platform that would match riders and drivers, while guaranteeing a ride back and exploiting spatial and temporal locality. It formalizes the CTSP to find a routing plan that maximizes ride sharing for a set of commute trips and proposes two exact algorithms for the CTSP: the REA and the BPA. The former exhaustively searches for feasible routes from all possible trip combinations, which are then supplied to a MIP which solves a set-partitioning problem. The latter uses column generation which applies a dynamic-programming algorithm to search for feasible routes with negative marginal costs on demand. A couple of clustering algorithms are also proposed to group trips based on the spatial proximity of the commuters' home locations or the spatio-temporal proximity of their trip itineraries to maintain problem tractability. The REA and the BPA are then used to optimally match commute trips from the real-world dataset of the city of Ann Arbor, Michigan.

Results of the computational experiments revealed that the BPA is better suited for problems with larger vehicle capacities, although they also revealed that there is very little benefit to utilizing vehicles with capacity greater than 4 in the CTSP as their effectiveness is mitigated by ride-duration limits of the drivers which restrict route length. On the other hand, the REA is found to be better suited for problems with large commuter counts, as it consistently produced results that are optimal or results that have optimality gaps that are often smaller than the BPA. The root-node solution of the BPA is also found to be a good heuristic for producing high-quality solutions in time-constrained scenarios, as it typically produces solutions with optimality gaps of $< 5\%$, even for larger problem instances, within a 10-minute time span.

When it is assumed that commuters are willing to shift their desired arrival and departure times by $\pm 10$ minutes and tolerate a 50% increase to their ride durations, the algorithms can produce optimal solutions for problems with up to 200 commuters and achieve high-quality results for problems with up to 400 commuters. When the maximum cluster size is set to 400, the results show the CTSP plans potentially reducing vehicle utilization by 57% and decreasing vehicle miles traveled by 46% at the cost of a 22% increase in average ride duration. The results thus highlight the significant potential and effectiveness of the CTSP in easing traffic and parking pressure on otherwise congested areas. Future work will be dedicated to further increasing the efficiency of the algorithms while making them more robust to changes in commuter schedules and additions of new customers to the commuting pool. Similarly, behavioral studies to determine the acceptance and adoption of such a car-pooling platform will be performed to understand how much of the theoretical potential can be achieved in various practical settings.

# CHAPTER 4

# The Flexible and Real-Time Commute Trip-Sharing Problems

## 4.1 Introduction

The CTSP was formalized in Chapter 3 in an effort to reduce peak-hour traffic congestion and parking utilization for urban areas. It seeks a car-pool routing plan that maximizes ride sharing for a set of commute trips. Each commuter makes two trip requests per day, one to the workplace and another back home. Each request has specific pickup and drop-off locations which must be visited in order, time windows describing allowable service times at each location, and a ride-duration limit. The routes must serve these trips exactly once according to their specifications while ensuring the capacity of the vehicles used is not exceeded. In addition to this, the vehicle drivers for any day are selected from the set of commuters (every commuter is assumed to possess a car and is therefore a potential driver); therefore, the set of drivers selected for the trips to the workplace must be identical to that for the return trips.

The CTSP is therefore a VRP with time-window, capacity, pairing, precedence, ride-duration, and driver constraints. It is a generalization of the VRPTW which is well known to be NP-hard (Savelsbergh 1985). It was solved in Chapter 2 using a three-stage approach which first clusters the commuters according to their residential locations, searches exhaustively for all feasible routes within each cluster, and then solves a MIP to optimize the selection of routes. Chapter 3 then introduced a branch-and-price algorithm to solve the problem that uses column generation to search for feasible routes on demand.

This work generalizes the CTSP by considering a setting in which trip schedules are known in advance, however there are uncertainties associated with the return-trip schedules as they occur later in the day. More specifically, it describes: (1) The Flexible Commute Trip-Sharing Problem (FCTSP) whereby the commuters are required to confirm their

return times by a fixed deadline and (2) the Real-Time Commute Trip-Sharing Problem (RT-CTSP) whereby the commuters confirm their return times in real time with some advance notice. Regardless of the problem variant, the routing plan must commit the day's drivers before the return times are confirmed. The challenge is therefore to ensure that the plan is robust, i.e., to ensure that the drivers can still cover the return trips despite the uncertainties in their schedules.

This chapter explores a scenario-sampling approach to handle these uncertainties. The method was first used by Bent and Van Hentenryck (2004) for the VRP with stochastic customers and then by Srour et al. (2018) to tackle temporal uncertainty in the VRPTW. It assumes knowledge of probability distributions for every commuter describing the likelihood of their return-trip times which can be sampled to obtain potential scenarios. This work incorporates the method into a multi-stage framework, whereby the first stage optimizes a plan for several scenarios while subsequent stages re-optimize the return-trip plan when additional information becomes available, i.e., just after the deadline for the FCTSP or at a regular frequency for the RT-CTSP which re-optimizes batches of trips with confirmed return times.

The approach is evaluated on the real-world dataset of commute trips from the city of Ann Arbor, Michigan. The results show that the method produces plans that become more robust as the number of sampled scenarios increases. Unfortunately, the increase in robustness comes at the price of using more vehicles. A method is therefore proposed to evaluate the trade-off between plan robustness and vehicle reduction.

The rest of this chapter is organized as follows. Section 4.2 summarizes the mathematical formulation of the CTSP and the column-generation algorithm from Chapter 3 for solving it. Section 4.3 then describes in detail the problem settings for the FCTSP and RT-CTSP together with their optimization algorithms. Section 4.4 reports the experimental setup and the results of the computational evaluations. Finally, Section 4.5 provides some concluding thoughts.

## 4.2 The Column-Generation Algorithm for the CTSP

The CTSP formulation uses $\Omega^+$ and $\Omega^-$ to denote the set of all feasible inbound and outbound routes to serve a set of commuters $\mathcal{C}$ on any given day. Binary variable $X_r$ indicates whether route $r \in \Omega^+ \cup \Omega^-$ is used in the optimal plan. The problem is defined by (4.1)–(4.5). Objective function (4.1) minimizes the number of routes used. Constraints (4.2) and (4.3) enforce coverage of each rider's inbound and outbound trips by exactly one route each, while constraints (4.4) ensure the sets of drivers for inbound and outbound routes are

identical.

$$\min \quad \sum_{r \in \Omega^+ \cup \Omega^-} X_r \tag{4.1}$$

$$\text{s.t.} \quad \sum_{r \in \Omega^+ : c \in \mathcal{C}_r} X_r = 1 \qquad \forall c \in \mathcal{C} \tag{4.2}$$

$$\sum_{r \in \Omega^- : c \in \mathcal{C}_r} X_r = 1 \qquad \forall c \in \mathcal{C} \tag{4.3}$$

$$\sum_{r \in \Omega^+ : D_r = c} X_r - \sum_{\hat{r} \in \Omega^- : D_{\hat{r}} = c} X_{\hat{r}} = 0 \qquad \forall c \in \mathcal{C} \tag{4.4}$$

$$X_r \in \{0, 1\} \qquad \forall r \in \Omega^+ \cup \Omega^- \tag{4.5}$$

Section 2.2.2 proposed an REA to possibly enumerate $\Omega^+ \cup \Omega^-$. The REA considers all commuter combinations of size $k \leq K$ (of which there are $\mathcal{O}(n^K)$ combinations in total, where $n = |\mathcal{C}|$ is the problem size), and for each $k$-combination, it considers each of the $k$ commuters as the potential driver of a route. Finally for each driver considered, it enumerates the $(2(k-1))!/(2^{k-1})$ possible location permutations for the route and checks the feasibility of each using the $feasible$ function described in Section 1.1.2. Section 2.2.2 showed that for a fixed vehicle capacity $K$, $|\Omega^+ \cup \Omega^-| = \mathcal{O}(n^K)$. Unfortunately, this enumeration procedure is only computationally practical for small $n$ and $K$.

For operational settings and problem instances whereby application of the REA is too expensive (such as the problem instances considered in this work), Chapter 3 proposed a column-generation algorithm which only considers routes with negative marginal costs that could potentially improve the model's objective function. The algorithm leverages the shadow prices of the constraints of an RMP—the linear relaxation of the original problem defined on a subset $\Omega^{+\prime} \cup \Omega^{-\prime}$ of all feasible routes—to calculate the reduced costs of routes. A PSP is executed alternately with the RMP to search for new routes with negative reduced costs which are then added to $\Omega^{+\prime} \cup \Omega^{-\prime}$. The RMP and the PSP are solved repeatedly until the PSP is unable to find any new route with negative reduced cost, at which point the optimal objective value of the RMP *converges* to the optimal objective value $z^*$ of the linear relaxation of the original problem. An integer solution is then obtained by solving the RMP as a MIP, and its quality is evaluated by calculating an optimality gap which uses $z^*$ as the primal lower bound.

The PSP considers each rider $c \in \mathcal{C}$ as the driver of an inbound route $r_c^+$ and an outbound route $r_c^-$, searches for such routes with minimum reduced costs, and adds them to $\Omega^{+\prime} \cup \Omega^{-\prime}$ should the costs be negative. The routes are found by first constructing a pair

of graphs $\mathcal{G}_c^+ = (\mathcal{N}_c^+, \mathcal{A}_c^+)$ and $\mathcal{G}_c^- = (\mathcal{N}_c^-, \mathcal{A}_c^-)$ for each driver $c$ whose nodes $\mathcal{N}_c^+$ and $\mathcal{N}_c^-$ represent the locations (the origins and destinations) of all inbound and outbound trips respectively and whose edges $\mathcal{A}_c^+$ and $\mathcal{A}_c^-$ represent location pairs that satisfy a priori feasibility constraints. Edge costs are calculated using the dual optimal solution of the RMP so that the total cost of any path from $o_c$ to $d_c$ is equivalent to the path's reduced cost. A wait-time relaxation algorithm based on the label-setting, dynamic-programming algorithm by Desrochers (1988) is then used to search for the minimum-cost feasible route from $o_c$ to $d_c$ from each graph. The algorithm searches for a preliminary feasible route with relaxed wait times and then verifies its feasibility with the inclusion of wait times using the $feasible$ function. Infeasible routes are added to a set of forbidden paths whose members are prevented from being discovered in subsequent runs, and the algorithm is executed repeatedly until a feasible solution is found. Complete details of the wait-time relaxation algorithm are provided in Section 3.4.3.

## 4.3 Robust Planning for the FCTSP and the RT-CTSP

In a practical setting for the CTSP, on any day, each commuter would make her trip request in the morning, specifying the desired arrival time at her workplace and the expected departure time for her return trip. The ride-sharing platform would then generate an optimal routing plan based on these times which consists of the day's designated drivers, the passengers they need to cover in their inbound and outbound routes, and the corresponding routes they need to take.

This work takes a step further by considering a setting in which the desired arrival times are certain as they occur in the morning, soon after the requests are made. However, the departure times might change due to unforeseen events occurring later in the day. The FCTSP considers a setting whereby the departure times are confirmed by a fixed deadline, whereas the RT-CTSP considers a more dynamic setting whereby the departure times are confirmed in real time with some advance notice. Regardless of the problem setting, an inbound routing plan which commits drivers for the day needs to be generated before the departure times can be confirmed. Therefore, *the driver assignment needs to be robust to ensure that they can cover as many return trips as possible, as the uncovered trips will need to be served by an external, more expensive resource.*

Most works on robust planning for transportation scheduling focus on addressing demand uncertainty. For instance, Serra et al. (2019) proposed a two-stage stochastic programming formulation for handling uncertain passengers in the integrated last-mile transportation problem (Raghunathan et al. 2018a,b), a problem which considers a fleet of shared

vehicles working in concert with a mass transit service to provide last-mile passenger transportation. To our knowledge, Srour et al. (2018) were the first to tackle service-time uncertainty in the VRPTW. They utilized scenario sampling, a method which assumes the differences between confirmed and forecast service times of every customer are random variables whose distributions can be gleaned from historical data. A sampled scenario is obtained by drawing presumed service times from every customer distribution, and the stochastic information from several sampled scenarios is leveraged by deriving a routing plan for each scenario and selecting the plan that most resembles every other plan using a consensus metric.

This work incorporates scenario sampling into a two-stage approach to produce robust plans for the FCTSP and the RT-CTSP. The first stage optimizes selection of drivers and inbound routes by solving a model that optimizes route selection for a single inbound scenario and a set of sampled outbound scenarios simultaneously. The second stage re-optimizes the outbound routing plan once the departure times have been confirmed by a deadline (FCTSP) or re-optimizes the outbound plan in a rolling-horizon approach as the departure times are progressively confirmed over time (RT-CTSP).

### 4.3.1 Stage 1: Optimizing Selection of Drivers and Inbound Routes

In the first stage, a model which selects drivers for the day together with their inbound routes is optimized. A single inbound scenario is derived from the commuters' desired arrival times, while a set of outbound scenarios is generated either by using the expected departure times to obtain a scenario where the commuters don't change their return schedules or by sampling the departure-time distributions of every commuter to obtain one where the commuters do. Let $\mathcal{S}$ denote the set of outbound scenarios and $\Omega_s^-$ denote the set of all feasible outbound routes for scenario $s$. The model is defined in terms of a binary variable $X_r$ which indicates whether a route $r$ is selected for the optimal plan. The model is defined by (4.6)–(4.10). Objective function (4.6) minimizes the total number of selected routes. Constraints (4.7) enforce coverage of each commuter's inbound trip by exactly one route, while constraints (4.8) do the same for each commuter's outbound trip in each scenario $s \in \mathcal{S}$. Finally, constraints (4.9) ensure the set of drivers selected for the inbound trips is identical to that for each outbound scenario $s \in \mathcal{S}$.

$$\min \quad \sum_{r\in\Omega^+} X_r + \sum_{s\in\mathcal{S}} \sum_{r\in\Omega_s^-} X_r \tag{4.6}$$

$$\text{s.t.} \quad \sum_{r\in\Omega^+:c\in\mathcal{C}_r} X_r = 1 \qquad \forall c \in \mathcal{C} \tag{4.7}$$

$$\sum_{r\in\Omega_s^-:c\in\mathcal{C}_r} X_r = 1 \qquad \forall s \in \mathcal{S}, \forall c \in \mathcal{C} \tag{4.8}$$

$$\sum_{r\in\Omega^+:D_r=c} X_r - \sum_{\hat{r}\in\Omega_s^-:D_{\hat{r}}=c} X_{\hat{r}} = 0 \qquad \forall s \in \mathcal{S}, \forall c \in \mathcal{C} \tag{4.9}$$

$$X_r \in \{0,1\} \qquad \forall s \in \mathcal{S}, \forall r \in \Omega^+ \cup \Omega_s^- \tag{4.10}$$

A column-generation algorithm similar to that used for the original CTSP is utilized to obtain a high-quality solution for the model. An RMP is first introduced as the linear relaxation of the model defined on a subset of all feasible routes, $\{\Omega^{+\prime} \cup \Omega_s^{-\prime} : s \in \mathcal{S}\}$. Let $\pi_c^+$, $\pi_{c,s}^-$, and $\sigma_{c,s}$ denote the optimal duals of constraints (4.7), (4.8), and (4.9) of the RMP respectively. The reduced cost of an inbound route $r^+$ is then given by:

$$rc_{r^+} = 1 - \sum_{c\in\mathcal{C}_{r^+}} \pi_c^+ - \sum_{s\in\mathcal{S}} \sigma_{D_{r^+},s} \tag{4.11}$$

while that of an outbound route for scenario $s$, $r_s^-$, is given by:

$$rc_{r_s^-} = 1 - \sum_{c\in\mathcal{C}_{r_s^-}} \pi_{c,s}^- + \sigma_{D_{r_s^-},s} \tag{4.12}$$

A PSP is then composed to find new routes with negative reduced costs. The PSP considers each rider $c \in \mathcal{C}$ as the driver of an inbound route $r_c^+$ and $|\mathcal{S}|$ outbound routes $\{r_{c,s}^- : s \in \mathcal{S}\}$. For each such route, the PSP finds one with minimum reduced cost and then selects those with negative reduced costs to augment $\{\Omega^{+\prime} \cup \Omega_s^{-\prime} : s \in \mathcal{S}\}$. To find these routes, a complete inbound graph $\mathcal{G}_c^+ = (\mathcal{N}_c^+, \mathcal{A}_c^+)$ and $|\mathcal{S}|$ complete outbound graphs $\{\mathcal{G}_{c,s}^- = (\mathcal{N}_{c,s}^-, \mathcal{A}_{c,s}^-) : s \in \mathcal{S}\}$ are first constructed for each driver $c$. The nodes in $\mathcal{N}_c^+$ represent all inbound origins $\mathcal{O}^+$ and destinations $\mathcal{D}^+$, whereas those in $\mathcal{N}_{c,s}^-$ represent all outbound origins $\mathcal{O}_s^-$ and destinations $\mathcal{D}_s^-$. A time window $[a_i, b_i]$ is associated with each inbound node $i \in \mathcal{O}^+ \cup \mathcal{D}^+$ based on the requested trip times and similarly a time window $[a_{i,s}, b_{i,s}]$ is associated with each outbound node $i \in \mathcal{O}_s^- \cup \mathcal{D}_s^-$ based on the trip times for scenario $s$. Edge costs are defined so that the total cost of any path from $o_c$ to $d_c$ is equivalent to the path's reduced cost. Let $c_{(i,j)}$ denote the cost of edge $(i,j)$ and $\delta^+(i)$

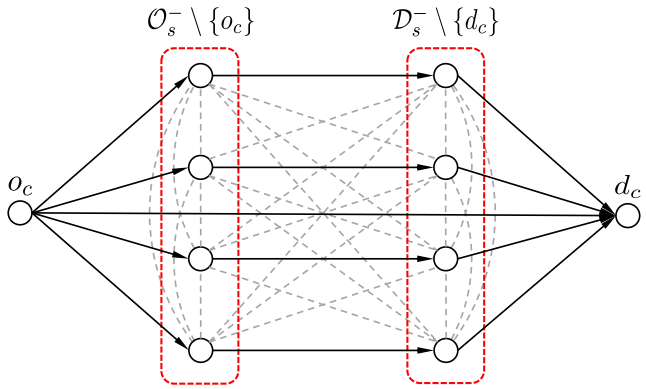



Figure 4.1: Graph $\mathcal{G}_{c,s}^-$ After Edge Elimination (Each Dotted Line Represents a Pair of Bidirectional Edges).

denote the set of outgoing edges of node $i$. Costs of edges $(i,j) \in \mathcal{A}_c^+$ are then given by:

$$c_{(i,j)} = \begin{cases} 1 - \pi_i^+ - \sum_{s \in \mathcal{S}} \sigma_{c,s} & \forall (i,j) \in \delta^+(o_c) \\ -\pi_i^+ & \forall i \in \mathcal{O}^+ \setminus \{o_c\},\ \forall (i,j) \in \delta^+(i) \\ 0 & \forall i \in \mathcal{D}^+,\ \forall (i,j) \in \delta^+(i) \end{cases} \tag{4.13}$$

while those of edges $(i,j) \in \mathcal{A}_{c,s}^-$ are given by:

$$c_{(i,j)} = \begin{cases} 1 - \pi_{i,s}^- + \sigma_{c,s} & \forall (i,j) \in \delta^+(o_c) \\ -\pi_{i,s}^- & \forall i \in \mathcal{O}_s^- \setminus \{o_c\},\ \forall (i,j) \in \delta^+(i) \\ 0 & \forall i \in \mathcal{D}_s^-,\ \forall (i,j) \in \delta^+(i) \end{cases} \tag{4.14}$$

A priori feasibility constraints proposed by Dumas et al. (1991), Cordeau (2006), and Hasan et al. (2020) —listed in Section 3.4.2—are then applied to all edges to identify and eliminate those that cannot belong to any feasible route. Figure 4.1 provides a sketch of $\mathcal{G}_{c,s}^-$ after edge elimination.

The wait-time relaxation algorithm described in Section 3.4.3 is then applied to find the minimum-cost feasible route from $o_c$ to $d_c$ from each graph, and the routes with negative costs are added to $\{\Omega^{+\prime} \cup \Omega_s^{-\prime} : s \in \mathcal{S}\}$. The RMP and the PSP are solved repeatedly until the RMP converges. An upper and a lower bound to $z^*$ are maintained for this purpose. The upper bound is given by the optimal objective value of the RMP after each iteration, $z_{\text{RMP}}$, while the lower bound $z_{\text{LB}}$ is calculated using the method by Farley (1990), where $z_{\text{LB}} = z_{\text{RMP}}/(1 - rc^*)$ and $rc^*$ denotes the smallest route reduced cost discovered in the PSP. Since constraints (4.9) restrict the objective value of any integer solution to be an integer multiple of $\beta = 1 + |\mathcal{S}|$, the column-generation procedure is terminated when

$\beta\lceil z_{\text{RMP}}/\beta\rceil - z_{\text{LB}} < \beta$.

After convergence, an integer solution is obtained by solving the RMP as a MIP, and its quality is assessed by calculating its optimality gap. Let $z_{\text{MIP}}$ denote the objective value of the MIP solution. The optimality gap is then given by $(z_{\text{MIP}} - z_{\text{RMP}})/z_{\text{MIP}}$ as $z_{\text{RMP}}$ is the primal lower bound to $z_{\text{MIP}}$. The solution of the first stage is given by the set of selected inbound routes $\hat{\mathcal{Z}} = \{r \in \Omega^{+\prime} : X_r = 1\}$ and their corresponding drivers $\hat{\mathcal{D}} = \{D_r : r \in \hat{\mathcal{Z}}\}$.

### 4.3.2 The FCTSP

The FCTSP considers a second stage that re-optimizes the outbound routing plan once the departure times have been confirmed by a fixed deadline. Let $\Omega^-_{\hat{\mathcal{D}}}$ be the set of all feasible outbound routes with drivers from $\hat{\mathcal{D}}$, i.e., $\Omega^-_{\hat{\mathcal{D}}} = \{r \in \Omega^- : D_r \in \hat{\mathcal{D}}\}$. The FCTSP model is defined in terms of two binary variables: $X_r$ to indicate the selection of route $r \in \Omega^-_{\hat{\mathcal{D}}}$ and $Y_c$ to indicate if rider $c$ cannot be covered in the outbound routing plan. The model is defined by (4.15)–(4.18). The objective function minimizes the number of uncovered riders and constraints (4.16) ensure $Y_c$ is set to 1 if rider $c$ cannot be covered by any outbound route.

$$\min \quad \sum_{c \in \mathcal{C}} Y_c \tag{4.15}$$

$$\text{s.t.} \quad \sum_{r \in \Omega^-_{\hat{\mathcal{D}}} : c \in \mathcal{C}_r} X_r + Y_c = 1 \qquad \forall c \in \mathcal{C} \tag{4.16}$$

$$X_r \in \{0,1\} \qquad \forall r \in \Omega^-_{\hat{\mathcal{D}}} \tag{4.17}$$

$$Y_c \in \{0,1\} \qquad \forall c \in \mathcal{C} \tag{4.18}$$

This model is solved using a column-generation algorithm similar to that used in the first stage. The key difference here is that only outbound routes driven by riders in $\hat{\mathcal{D}}$ need to be generated in this model. Letting $\lambda_c$ denote the optimal dual of constraints (4.16) of the model's RMP, the reduced cost of an outbound route $r$ is given by:

$$rc_r = -\sum_{c \in \mathcal{C}_r} \lambda_c \tag{4.19}$$

The PSP of this model finds routes with negative reduced costs by first constructing a graph $\hat{\mathcal{G}}^-_c = (\hat{\mathcal{N}}^-_c, \hat{\mathcal{A}}^-_c)$ for each driver $c \in \hat{\mathcal{D}}$, finding the route from $o_c$ to $d_c$ with minimum

reduced cost from each $\hat{\mathcal{G}}_c^-$, and adding the route to $\Omega_{\hat{\mathcal{D}}}^{-\prime}$ if its cost is negative. The nodes $\hat{\mathcal{N}}_c^-$ consist of all outbound origins $\mathcal{O}^-$ and destinations $\mathcal{D}^-$, and each node $i \in \mathcal{O}^- \cup \mathcal{D}^-$ has associated with it a time window $[a_i, b_i]$ based on the *confirmed outbound trip times*. The costs of edges $(i,j) \in \hat{\mathcal{A}}_c^-$ are given by:

$$c_{(i,j)} = \begin{cases} -\lambda_i & \forall i \in \mathcal{O}^- \\ 0 & \forall i \in \mathcal{D}^- \end{cases} \tag{4.20}$$

so that the total cost of any path from $o_c$ to $d_c$ is equivalent to $rc_r$.

The same wait-time relaxation algorithm from the first stage is also used to find the minimum-cost feasible route from $o_c$ to $d_c$ for each $\hat{\mathcal{G}}_c^-$. The RMP and the PSP are solved repeatedly until the PSP cannot find any new route with negative reduced cost, after which the RMP is solved as a MIP to obtain an integer solution.

### 4.3.3 The RT-CTSP

The RT-CTSP is tailored for a setting where the departure times are continuously confirmed by the riders over time. It requires each rider $i$ to confirm her actual departure time, $dt_i^{\text{actual}}$, in advance by at least a time interval $\Delta_{\text{lead}}$. In other words, $dt_i^{\text{actual}}$ is confirmed at time $ct_i \leq dt_i^{\text{actual}} - \Delta_{\text{lead}}$. Interval $\Delta_{\text{lead}}$ will be referred to henceforth as the lead time, and it is assumed to be identical for every rider.

A rolling-horizon approach which executes the FCTSP optimization algorithm from Section 4.3.2 once every $\Delta_{\text{opt}}$, where $\Delta_{\text{opt}}$ is a fixed time interval, is proposed for this setting. Let $ot_k$ denote the time of the $k^{\text{th}}$ optimization run of this approach and $\mathcal{Z}_k$ be its solution (i.e., its set of selected routes). Each optimization run includes a batch of trips whose departure times have been confirmed, i.e., the trips of riders $\{i \in \mathcal{C} : ct_i \leq ot_k\}$. At the same time, a different set of trips is excluded from the $k^{\text{th}}$ run. These are the trips whose departure-time windows have expired, i.e., the trips of riders $\{i \in \mathcal{C} : b_{o_i} \leq ot_k\}$, and the trips covered by the routes from $\mathcal{Z}_{k-1}$ that have already departed. Letting $st_r$ denote the starting time of route $r$, the latter set of excluded trips are those of riders $\{i \in \mathcal{C}_r : r \in \mathcal{Z}_{k-1}, st_r \leq ot_k\}$.

The scheme is executed until each rider $i \in \mathcal{C}$ is either served by a departed route or has her departure-time window expire without being served by any route. The riders that are not served by any route will have to resort to an external resource to cover their return trips. The rolling-horizon algorithm is summarized in Algorithm 2. The riders that are not served by any route are stored in a set of uncovered riders $\mathcal{U}$, whereas the routes that have departed throughout the execution of the algorithm are stored in $\mathcal{Z}_{\text{rolling-horizon}}$ which represents the

**Algorithm 2** Rolling-Horizon Optimization of Outbound Routing Plan

$k \leftarrow 1$
$ot_k \leftarrow 0$, $\mathcal{Z}_{k-1} \leftarrow \emptyset$
$\mathcal{Z}_{\text{rolling-horizon}} \leftarrow \emptyset$, $\mathcal{U} \leftarrow \emptyset$
**while** $\mathcal{C} \neq \emptyset$ **do**
    $\mathcal{Z}_{\text{rolling-horizon}} \leftarrow \mathcal{Z}_{\text{rolling-horizon}} \cup \{r \in \mathcal{Z}_{k-1} : st_r \leq ot_k\}$
    $\mathcal{C} \leftarrow \mathcal{C} \setminus \{i \in \mathcal{C}_r : r \in \mathcal{Z}_{\text{rolling-horizon}}\}$
    $\mathcal{U} \leftarrow \mathcal{U} \cup \{i \in \mathcal{C} : b_{o_i} \leq ot_k\}$
    $\mathcal{C} \leftarrow \mathcal{C} \setminus \mathcal{U}$
    $\mathcal{C}_k \leftarrow \{i \in \mathcal{C} : ct_i \leq ot_k\}$
    $\mathcal{Z}_k \leftarrow$ Solution of FCTSP re-optimization on outbound trips of riders in $\mathcal{C}_k$
    $k \leftarrow k + 1$
    $ot_k \leftarrow ot_{k-1} + \Delta_{\text{opt}}$
**return** $\mathcal{Z}_{\text{rolling-horizon}}$, $\mathcal{U}$

final solution of the approach.

Note that in this algorithm, the routes that have departed are considered finalized and therefore, they are never modified post departure. This is done because, as was revealed in Chapter 3, the routes of the CTSP are typically very short as each driver can only pick up and drop off one set of passengers before having to end the route to ensure satisfaction of the time-window constraints at her trip destination and her ride-duration limit. There is therefore very little to be gained from the redirection of these short routes post departure. This, combined with the potential inconvenience that may be induced on the drivers by the route redirections, led to the decision of not modifying the routes once they have departed.

## 4.4 Computational Results

### 4.4.1 The Experimental Setting

The algorithms are evaluated on trips from the Ann Arbor commute-trip dataset. The experiments focus on the trips of commuters living within the city limits, which amount to approximately 2,400 trips per weekday. To maintain tractability, the trips are partitioned into smaller problem instances by clustering the commuters into clusters of size $n \approx 300$ using the spatial clustering algorithm described in Section 2.2.1 and only considering ride sharing intra-cluster.

Similar to Jaw et al. (1986), Cordeau and Laporte (2003b), and Cordeau (2006), this work assumes on any day, each commuter $c$ would specify a desired arrival time at the destination of her inbound trip, $at_c^+$, and a desired departure time at the origin of her outbound

trip, $dt_c^-$. It also assumes that each would tolerate a shift of $\pm\Delta$ to the desired times. Therefore, time windows of $[a_{d_c}, b_{d_c}] = [at_c^+ - \Delta, at_c^+ + \Delta]$ and $[a_{o_c}, b_{o_c}] = [dt_c^- - \Delta, dt_c^- + \Delta]$ are associated with the destinations of inbound trips and origins of outbound trips respectively. Consequently, time windows of $[a_{o_c}, b_{o_c}] = [a_{d_c} - \zeta_{o_c} - L_{t_c}, b_{d_c} - \zeta_{o_c} - \tau_{(o_c,d_c)}]$ and $[a_{d_c}, b_{d_c}] = [a_{o_c} + \zeta_{o_c} + \tau_{(o_c,d_c)}, b_{o_c} + \zeta_{o_c} + L_{t_c}]$ are assigned to the origins of inbound trips and destinations of outbound trips respectively. Similar to Hunsaker and Savelsbergh (2002), each rider $c$ is also assumed to be willing to tolerate an $R\%$ extension to her direct-ride duration $\tau_{(o_c,d_c)}$; therefore $L_{t_c} = (1 + R)\tau_{(o_c,d_c)}$.

A Laplace distribution is fit to the historical departure times of each commuter using maximum-likelihood estimation and the distributions are sampled to generate an outbound scenario $s$. The Laplace distribution is selected due to its suitability for modeling phenomena with heavy tails or those with higher peaks than the normal distribution. It is used in this work based on the assumption that the commuters are very likely to depart close to a standard departure time, however there may be exceptions where they are delayed or need to leave early. For a selected day, the inbound scenario for a set of commuters $\mathcal{C}$ is obtained using their arrival times from the dataset. The departure times from the dataset, however, are treated as expected departure times, and they are used to construct an expected outbound scenario, $s_{\text{expected}}$. The actual scenario is simulated by first selecting a fraction $f$ of the commuters uniformly at random and then sampling their departure-time distributions. The set of selected commuters, $\mathcal{C}_f$, represents those who had to change their return schedules. The actual departure times of the remaining commuters in $\mathcal{C} \setminus \mathcal{C}_f$ stay at their expected values. Every set of outbound scenarios $\mathcal{S}$ contains $s_{\text{expected}}$ to ensure that every return trip can be covered when there are absolutely no schedule changes, as $s_{\text{expected}}$ corresponds to the scenario where $f = 0.0$. Therefore, when $|\mathcal{S}| = 1$, $\mathcal{S} = \{s_{\text{expected}}\}$. The algorithms are implemented in C++ and they invoke Gurobi 7.5.1 to solve the LPs and the MIPs. All experiments use $K = 4$, $\Delta = 10$ mins, and $R = 0.50$, and they are conducted on a high-performance computing cluster with 12 cores of a 2.5 GHz Intel Xeon E5-2680v3 processor and 32 GB of RAM.

### 4.4.2 The Impact of the Number of Scenarios on the Vehicle Count

The first set of experiments vary the number of outbound scenarios, $|\mathcal{S}|$, in the first stage to measure the "price" of robustness, i.e., its impact on the number of vehicles. Table 4.1 summarizes the results from solving the first-stage model on eight clusters for $|\mathcal{S}| \in \{1, 4, 8, 12, 16, 20\}$. Since the experiments for $|\mathcal{S}| > 1$ involve random sampling, each is repeated 20 times and the table lists the average results from the 20 runs. Its first three

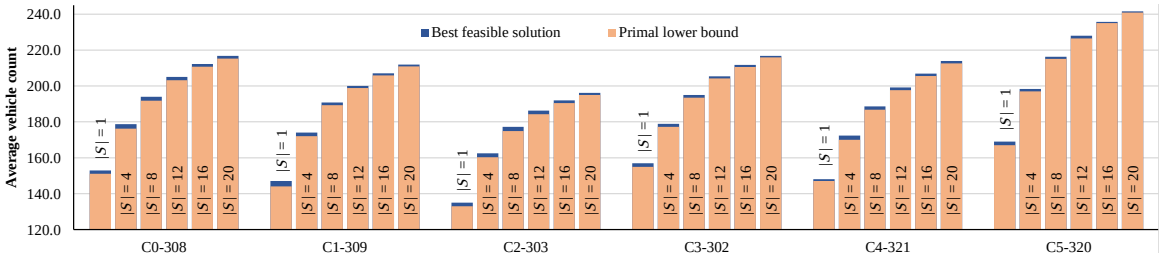



Figure 4.2: Average Vehicle Counts from the First-Stage Model for Clusters C0-308, C1-309, C2-303, C3-302, C4-321, and C5-320.

columns list the cluster IDs, the cluster sizes, and the number of outbound scenarios used. The next three display the model's results in terms of the average number of columns generated, the average vehicle count, and the average optimality gap of the solution. The final three columns show the average times spent on solving the RMP, the MIP, and the problem instance as a whole. Note that a 60 s time limit was placed on the MIP solver, and the vehicle counts reported are from the best feasible solutions. The time limit allows each problem instance to be solved in $< 15$ minutes, and despite the time limit, every solution produced an optimality gap of $< 3\%$. Figure 4.2 summarizes the effect of increasing $|\mathcal{S}|$ on the first six clusters (the results of the remaining two are consistent). The key observation here is that the vehicle count increases with $|\mathcal{S}|$, a result which can be attributed to the model having to cater to more outbound scenarios.

### 4.4.3 The FCTSP

The second set of results considers the FCTSP on the same instances with $f \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ : $f = 0.0$ models a scenario whereby none of the commuters change their departure times, whereas $f = 1.0$ models one in which every commuter does. The results are summarized in Table 4.2. Similar to Table 4.1, the first three columns display the cluster IDs, the cluster sizes, and the outbound scenario counts, while the next lists the vehicle count results of the first-stage model. The final six columns show the number of uncovered riders resulting from the re-optimized outbound plans of the FCTSP for each $f$ value. Since the experiments for $f > 0.0$ involve the random selection of $\mathcal{C}_f$ and the random sampling of their departure times, the experiment for each $f$ value is repeated 25 times and the table lists the average results from the 25 runs. The computation times are not shown here as every instance was solved in less that 15 s. Figure 4.3 illustrates how increasing the number of sampled scenarios improves the robustness of the outbound plan by showing the average number of uncovered riders for cluster C1-309 for $f \in [0.0, 1.0]$. As expected, all riders are covered when $f = 0.0$ as every set $\mathcal{S}$ contains $s_{\text{expected}}$. Moreover, the number

Table 4.1: First-Stage Optimization Results for $|\mathcal{S}| \in \{1, 4, 8, 12, 16, 20\}$.

| Cluster ID | Cluster size | Scenario count | Average column count | Average vehicle count | Average optimality gap (%) | Average wall time (s) | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | RMP | MIP | Total |
| C0-308 | 308 | 1 | 4883.0 | 153.0 | 1.31 | 76.3 | 3.8 | 80.1 |
| | | 4 | 10081.0 | 178.8 | 1.46 | 159.8 | 49.1 | 208.9 |
| | | 8 | 16695.2 | 194.1 | 1.16 | 277.1 | 46.6 | 323.7 |
| | | 12 | 23013.4 | 205.1 | 0.93 | 392.3 | 30.4 | 422.8 |
| | | 16 | 29373.8 | 212.3 | 0.75 | 510.5 | 19.5 | 530.0 |
| | | 20 | 35840.6 | 216.8 | 0.72 | 629.9 | 17.9 | 647.9 |
| C1-309 | 309 | 1 | 5398.0 | 147.0 | 2.04 | 90.1 | 3.2 | 93.3 |
| | | 4 | 11459.5 | 174.1 | 1.12 | 188.8 | 43.9 | 232.7 |
| | | 8 | 18684.2 | 190.8 | 0.79 | 297.0 | 25.1 | 322.1 |
| | | 12 | 25834.0 | 200.1 | 0.63 | 418.9 | 17.2 | 436.1 |
| | | 16 | 32887.2 | 207.1 | 0.55 | 541.0 | 16.8 | 557.8 |
| | | 20 | 40018.5 | 211.9 | 0.43 | 658.4 | 8.4 | 666.8 |
| C2-303 | 303 | 1 | 5508.0 | 135.0 | 1.48 | 112.9 | 34.4 | 147.3 |
| | | 4 | 12066.0 | 162.5 | 1.33 | 256.8 | 57.0 | 313.8 |
| | | 8 | 20178.2 | 177.2 | 1.24 | 361.9 | 53.1 | 415.0 |
| | | 12 | 27980.1 | 186.2 | 0.99 | 469.9 | 48.5 | 518.4 |
| | | 16 | 35783.6 | 192.1 | 0.83 | 607.4 | 48.1 | 655.5 |
| | | 20 | 43499.9 | 196.2 | 0.61 | 753.7 | 42.9 | 796.6 |
| C3-302 | 302 | 1 | 4343.0 | 157.0 | 1.27 | 59.4 | 2.2 | 61.6 |
| | | 4 | 9102.5 | 179.0 | 1.00 | 143.1 | 38.4 | 181.5 |
| | | 8 | 15134.6 | 195.1 | 0.79 | 252.7 | 26.0 | 278.7 |
| | | 12 | 21046.1 | 205.4 | 0.51 | 366.0 | 14.7 | 380.7 |
| | | 16 | 26973.7 | 211.7 | 0.52 | 468.0 | 11.0 | 478.9 |
| | | 20 | 32909.5 | 216.9 | 0.39 | 573.5 | 7.6 | 581.0 |
| C4-321 | 321 | 1 | 5828.0 | 148.0 | 0.68 | 88.2 | 60.0 | 148.2 |
| | | 4 | 12828.7 | 172.4 | 1.42 | 226.4 | 55.3 | 281.7 |
| | | 8 | 21312.7 | 188.6 | 0.98 | 371.4 | 47.7 | 419.1 |
| | | 12 | 29317.1 | 199.2 | 0.78 | 500.7 | 32.4 | 533.0 |
| | | 16 | 37375.4 | 206.9 | 0.68 | 649.2 | 25.5 | 674.7 |
| | | 20 | 45243.6 | 213.9 | 0.63 | 780.9 | 20.1 | 801.0 |
| C5-320 | 320 | 1 | 4457.0 | 169.0 | 1.18 | 66.4 | 6.7 | 73.0 |
| | | 4 | 9193.4 | 198.4 | 0.68 | 162.0 | 19.2 | 181.2 |
| | | 8 | 15239.4 | 216.3 | 0.56 | 286.5 | 8.0 | 294.5 |
| | | 12 | 21258.2 | 228.1 | 0.68 | 409.9 | 9.5 | 419.4 |
| | | 16 | 27342.9 | 235.8 | 0.32 | 532.7 | 4.2 | 536.9 |
| | | 20 | 33495.3 | 241.7 | 0.29 | 661.2 | 2.1 | 663.3 |
| C6-300 | 300 | 1 | 5479.0 | 137.0 | 1.46 | 83.5 | 43.2 | 126.8 |
| | | 4 | 11953.5 | 160.3 | 1.53 | 182.9 | 57.5 | 240.5 |
| | | 8 | 19708.7 | 177.3 | 1.10 | 291.4 | 40.8 | 332.2 |
| | | 12 | 27263.2 | 186.9 | 0.75 | 402.2 | 26.8 | 429.0 |
| | | 16 | 34790.7 | 194.5 | 0.62 | 512.7 | 22.6 | 535.3 |
| | | 20 | 42193.0 | 200.3 | 0.50 | 612.2 | 15.5 | 627.7 |
| C7-299 | 299 | 1 | 3952.0 | 165.0 | 0.61 | 61.9 | 12.2 | 74.1 |
| | | 4 | 8724.6 | 190.8 | 0.71 | 145.4 | 15.8 | 161.2 |
| | | 8 | 14866.7 | 207.6 | 0.48 | 252.1 | 19.5 | 271.6 |
| | | 12 | 20976.4 | 216.6 | 0.51 | 356.6 | 11.6 | 368.2 |
| | | 16 | 27013.2 | 223.4 | 0.40 | 468.3 | 9.6 | 477.9 |
| | | 20 | 33037.1 | 227.9 | 0.28 | 575.9 | 4.8 | 580.7 |

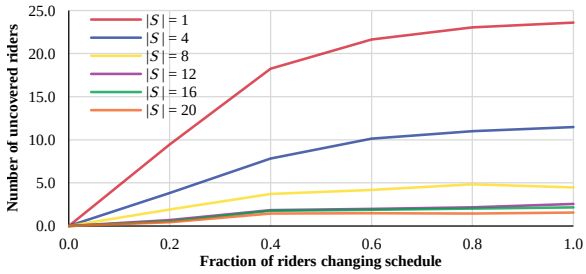



Figure 4.3: Average Number of Uncovered Riders from the FCTSP for Cluster C1-309.

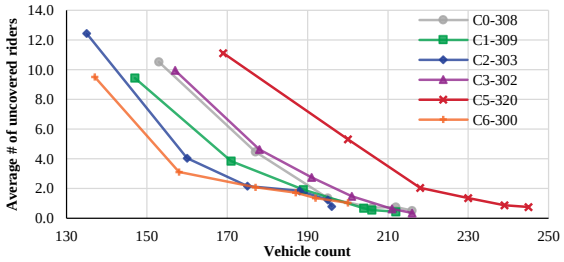



Figure 4.4: Average Number of Uncovered Riders and Vehicle Count Results of the FCTSP for Several Clusters with $|\mathcal{S}| \in \{1, 4, 8, 12, 16, 20\}$ and $f = 0.2$.

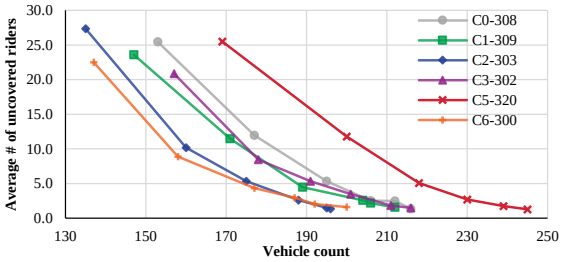



Figure 4.5: Average Number of Uncovered Riders and Vehicle Count Results of the FCTSP for Several Clusters with $|\mathcal{S}| \in \{1, 4, 8, 12, 16, 20\}$ and $f = 1.0$.

of uncovered riders generally increases with $f$ for any fixed $|\mathcal{S}|$, indicating the increasing challenge of accommodating progressively more changing schedules. However, for any fixed $f$, the average number of uncovered riders declines as $|\mathcal{S}|$ is increased, signifying an increase in plan robustness. The marginal benefits, however, diminish with increasing $|\mathcal{S}|$, and the benefits come at the price of increases in vehicle count as shown in Figure 4.2. This trade-off is further illustrated in Figures 4.4 and 4.5 which plot the average number of uncovered riders for several clusters against their corresponding vehicle counts for each $|\mathcal{S}| \in \{1, 4, 8, 12, 16, 20\}$ when $f = 0.2$ and $f = 1.0$ respectively. Each point in the figures represents the average number of uncovered riders and the vehicle count results for a specific $|\mathcal{S}|$ value and, for each curve, as $|\mathcal{S}|$ increases going from left to right, so do plan robustness and vehicle count, highlighting the trade-off between robustness and vehicle reduction.

### 4.4.4 The RT-CTSP

The next set of results considers the RT-CTSP on the same instances with $\Delta_{\text{opt}} = 10$ mins, $|\mathcal{S}| \in \{1, 12\}$, and $\Delta_{\text{lead}} \in \{30, 60, 90\}$ mins. Table 4.3 summarizes these results. The first four columns list the cluster IDs, the cluster sizes, the outbound scenario counts,

Table 4.2: FCTSP Results for $|\mathcal{S}| \in \{1, 4, 8, 12, 16, 20\}$ and $f \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$.

| Cluster ID | Cluster size | Scenario count | Vehicle count | Average # of uncovered riders | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | $f = 0.0$ | $f = 0.2$ | $f = 0.4$ | $f = 0.6$ | $f = 0.8$ | $f = 1.0$ |
| C0-308 | 308 | 1 | 153 | 0.0 | 10.5 | 17.1 | 21.3 | 24.8 | 25.4 |
| | | 4 | 177 | 0.0 | 4.5 | 8.1 | 10.0 | 10.9 | 12.0 |
| | | 8 | 195 | 0.0 | 1.4 | 3.2 | 4.0 | 4.8 | 5.3 |
| | | 12 | 206 | 0.0 | 0.7 | 1.8 | 2.1 | 2.8 | 2.5 |
| | | 16 | 212 | 0.0 | 0.8 | 1.3 | 1.5 | 2.2 | 2.5 |
| | | 20 | 216 | 0.0 | 0.5 | 1.0 | 1.2 | 1.4 | 1.4 |
| C0-308 | 308 | 1 | 153 | 0.0 | 10.5 | 17.1 | 21.3 | 24.8 | 25.4 |
| | | 4 | 177 | 0.0 | 4.5 | 8.1 | 10.0 | 10.9 | 12.0 |
| | | 8 | 195 | 0.0 | 1.4 | 3.2 | 4.0 | 4.8 | 5.3 |
| | | 12 | 206 | 0.0 | 0.7 | 1.8 | 2.1 | 2.8 | 2.5 |
| | | 16 | 212 | 0.0 | 0.8 | 1.3 | 1.5 | 2.2 | 2.5 |
| | | 20 | 216 | 0.0 | 0.5 | 1.0 | 1.2 | 1.4 | 1.4 |
| C2-303 | 303 | 1 | 135 | 0.0 | 12.4 | 20.1 | 25.8 | 27.2 | 27.4 |
| | | 4 | 160 | 0.0 | 4.0 | 7.7 | 9.8 | 10.0 | 10.2 |
| | | 8 | 175 | 0.0 | 2.2 | 3.8 | 5.2 | 5.3 | 5.4 |
| | | 12 | 188 | 0.0 | 1.8 | 2.2 | 2.2 | 2.3 | 2.6 |
| | | 16 | 195 | 0.0 | 1.2 | 1.5 | 1.4 | 1.4 | 1.5 |
| | | 20 | 196 | 0.0 | 0.8 | 1.2 | 1.1 | 1.1 | 1.4 |
| C3-302 | 302 | 1 | 157 | 0.0 | 10.0 | 17.4 | 19.3 | 21.9 | 20.9 |
| | | 4 | 178 | 0.0 | 4.6 | 8.1 | 7.3 | 8.6 | 8.4 |
| | | 8 | 191 | 0.0 | 2.8 | 4.8 | 4.6 | 5.5 | 5.3 |
| | | 12 | 201 | 0.0 | 1.5 | 3.5 | 2.8 | 3.9 | 3.5 |
| | | 16 | 211 | 0.0 | 0.6 | 2.2 | 1.7 | 2.3 | 1.8 |
| | | 20 | 216 | 0.0 | 0.4 | 1.5 | 1.0 | 1.9 | 1.5 |
| C4-321 | 321 | 1 | 148 | 0.0 | 9.3 | 18.2 | 22.4 | 24.0 | 25.0 |
| | | 4 | 169 | 0.0 | 5.2 | 9.4 | 11.4 | 11.4 | 12.8 |
| | | 8 | 184 | 0.0 | 3.0 | 4.8 | 7.0 | 6.2 | 7.7 |
| | | 12 | 195 | 0.0 | 1.4 | 2.6 | 3.8 | 3.6 | 4.6 |
| | | 16 | 203 | 0.0 | 1.0 | 1.9 | 3.2 | 2.5 | 3.2 |
| | | 20 | 211 | 0.0 | 0.8 | 1.6 | 2.4 | 2.0 | 2.1 |
| C5-320 | 320 | 1 | 169 | 0.0 | 11.1 | 20.5 | 26.7 | 26.8 | 25.5 |
| | | 4 | 200 | 0.0 | 5.3 | 10.2 | 12.1 | 11.8 | 11.8 |
| | | 8 | 218 | 0.0 | 2.0 | 4.8 | 5.8 | 5.4 | 5.1 |
| | | 12 | 230 | 0.0 | 1.4 | 2.8 | 3.3 | 3.1 | 2.7 |
| | | 16 | 239 | 0.0 | 0.9 | 1.5 | 1.8 | 1.9 | 1.8 |
| | | 20 | 245 | 0.0 | 0.8 | 1.2 | 1.3 | 1.4 | 1.3 |
| C6-300 | 300 | 1 | 137 | 0.0 | 9.5 | 17.6 | 23.5 | 22.5 | 22.5 |
| | | 4 | 158 | 0.0 | 3.1 | 6.8 | 9.1 | 9.2 | 8.9 |
| | | 8 | 177 | 0.0 | 2.1 | 4.5 | 5.2 | 4.9 | 4.4 |
| | | 12 | 187 | 0.0 | 1.7 | 3.2 | 3.9 | 3.9 | 3.0 |
| | | 16 | 192 | 0.0 | 1.4 | 1.7 | 2.6 | 2.5 | 2.0 |
| | | 20 | 200 | 0.0 | 1.0 | 1.3 | 1.8 | 2.0 | 1.6 |
| C7-299 | 299 | 1 | 165 | 0.0 | 8.6 | 17.4 | 21.7 | 23.4 | 24.0 |
| | | 4 | 192 | 0.0 | 3.9 | 7.3 | 8.0 | 8.0 | 9.4 |
| | | 8 | 211 | 0.0 | 1.6 | 2.8 | 3.3 | 4.7 | 3.8 |
| | | 12 | 218 | 0.0 | 1.8 | 2.8 | 3.0 | 3.7 | 3.7 |
| | | 16 | 227 | 0.0 | 0.8 | 1.5 | 1.2 | 2.4 | 1.6 |
| | | 20 | 228 | 0.0 | 0.6 | 1.4 | 1.1 | 2.1 | 1.4 |

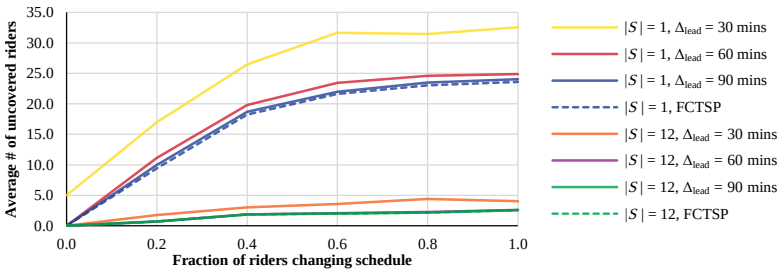



Figure 4.6: Average Number of Uncovered Riders from the RT-CTSP for Cluster C1-309.

and the lead time values, while the remaining six show results of the algorithm for each $f \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ in terms of the average number of uncovered riders obtained from repeating the experiment 25 times for each $f$ value. Figure 4.6 then illustrates the results for cluster C1-309 and it includes the results of the FCTSP for additional perspective. As expected, the robustness of the rolling-horizon method is never better than that of the FCTSP as the former only optimizes batches of trips at a time. However, the results of the former is only slightly worse than the latter when $|\mathcal{S}| = 12$, demonstrating the viability of the real-time approach. Its results also quickly approach those of the FCTSP as the lead time is increased.

### 4.4.5 Evaluating the Plan Robustness-Vehicle Reduction Trade-Off

Figure 4.7 reexamines the average uncovered riders-vehicle count curve of cluster C2-303 when $f = 1.0$ (taken from Figure 4.5), which is the worst-case scenario obviously. Each point on the curve represents a potential operating point for the ride-sharing platform, and the goal is to find the point with the best trade-off between plan robustness and vehicle reduction. Let $\Delta_u$ and $\Delta_v$ denote the change in the number of uncovered riders and the change in vehicle count respectively resulting from moving between two points along the curve, and let $c_u$ and $c_v$ be the cost per uncovered rider and the cost per unit of vehicle increase respectively. The marginal cost of moving between two points along the curve is therefore given by $\Delta_u c_u + \Delta_v c_v$. A negative marginal cost will result in a reduction to the operating cost, thus moving from one point on the curve to another is beneficial when $\Delta_u c_u + \Delta_v c_v < 0$. Rearranging the inequality results in $c_v/c_u < -\Delta_u/\Delta_v$, where $\Delta_u/\Delta_v$ is given by the slope of the curve. Therefore, it is beneficial to move right along the curve when the $c_v/c_u$ ratio is less then the negative of the curve's slope.

For instance, consider the leftmost curve segment in Figure 4.7 which has a slope of $-0.69$. Suppose the cost of an uncovered rider is given by the average price of using a ride-hailing service to cover a return trip, while the cost per unit of vehicle increase is

Table 4.3: RT-CTSP Results for $\Delta_{\text{opt}} = 10$ mins, $|\mathcal{S}| \in \{1, 12\}$, $\Delta_{\text{lead}} \in \{30, 60, 90\}$ mins, and $f \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$.

| Cluster ID | Cluster size | Scenario count | Lead time (mins) | Average # of uncovered riders | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | $f = 0.0$ | $f = 0.2$ | $f = 0.4$ | $f = 0.6$ | $f = 0.8$ | $f = 1.0$ |
| C0-308 | 308 | 1 | 30 | 9.0 | 17.8 | 24.2 | 28.7 | 33.0 | 33.2 |
| | | | 60 | 1.0 | 11.8 | 18.1 | 22.5 | 25.8 | 26.6 |
| | | | 90 | 0.0 | 11.1 | 17.4 | 21.6 | 25.2 | 25.7 |
| | | 12 | 30 | 0.0 | 1.7 | 2.9 | 3.5 | 4.8 | 4.6 |
| | | | 60 | 0.0 | 0.8 | 1.8 | 2.1 | 2.8 | 2.6 |
| | | | 90 | 0.0 | 0.7 | 1.8 | 2.1 | 2.8 | 2.5 |
| C1-309 | 309 | 1 | 30 | 5.0 | 17.0 | 26.4 | 31.7 | 31.5 | 32.6 |
| | | | 60 | 0.0 | 11.1 | 19.8 | 23.4 | 24.6 | 24.9 |
| | | | 90 | 0.0 | 10.0 | 18.7 | 22.0 | 23.5 | 24.0 |
| | | 12 | 30 | 0.0 | 1.8 | 3.0 | 3.6 | 4.4 | 4.0 |
| | | | 60 | 0.0 | 0.7 | 1.8 | 2.0 | 2.2 | 2.6 |
| | | | 90 | 0.0 | 0.7 | 1.8 | 2.0 | 2.2 | 2.6 |
| C2-303 | 303 | 1 | 30 | 6.0 | 19.6 | 28.2 | 33.7 | 34.9 | 35.6 |
| | | | 60 | 0.0 | 14.1 | 21.4 | 27.3 | 28.4 | 28.8 |
| | | | 90 | 0.0 | 13.0 | 20.5 | 26.4 | 27.6 | 27.9 |
| | | 12 | 30 | 3.0 | 4.3 | 4.6 | 4.9 | 5.2 | 6.4 |
| | | | 60 | 0.0 | 2.2 | 2.4 | 2.4 | 2.4 | 2.8 |
| | | | 90 | 0.0 | 1.9 | 2.2 | 2.2 | 2.3 | 2.6 |
| C3-302 | 302 | 1 | 30 | 3.0 | 16.4 | 24.7 | 26.5 | 28.5 | 28.1 |
| | | | 60 | 0.0 | 11.1 | 18.4 | 20.5 | 22.2 | 21.8 |
| | | | 90 | 0.0 | 10.4 | 17.6 | 19.5 | 22.0 | 21.0 |
| | | 12 | 30 | 0.0 | 2.9 | 5.4 | 5.1 | 6.3 | 5.4 |
| | | | 60 | 0.0 | 1.6 | 3.6 | 3.0 | 4.1 | 3.6 |
| | | | 90 | 0.0 | 1.5 | 3.5 | 2.8 | 3.9 | 3.5 |
| C4-321 | 321 | 1 | 30 | 6.0 | 17.3 | 27.0 | 31.5 | 32.7 | 34.1 |
| | | | 60 | 0.0 | 11.0 | 20.0 | 24.4 | 26.1 | 26.9 |
| | | | 90 | 0.0 | 10.0 | 18.8 | 23.0 | 24.7 | 25.6 |
| | | 12 | 30 | 2.0 | 4.4 | 5.6 | 6.3 | 6.0 | 8.0 |
| | | | 60 | 0.0 | 1.6 | 2.8 | 3.9 | 3.8 | 5.1 |
| | | | 90 | 0.0 | 1.4 | 2.6 | 3.8 | 3.6 | 4.7 |
| C5-320 | 320 | 1 | 30 | 1.0 | 16.3 | 27.5 | 32.1 | 32.7 | 31.6 |
| | | | 60 | 0.0 | 12.2 | 21.7 | 27.3 | 27.7 | 26.1 |
| | | | 90 | 0.0 | 11.6 | 20.8 | 26.7 | 27.0 | 25.6 |
| | | 12 | 30 | 1.0 | 2.8 | 4.0 | 4.9 | 4.4 | 3.8 |
| | | | 60 | 0.0 | 1.5 | 2.9 | 3.4 | 3.1 | 2.8 |
| | | | 90 | 0.0 | 1.4 | 2.8 | 3.3 | 3.1 | 2.7 |
| C6-300 | 300 | 1 | 30 | 8.0 | 16.8 | 26.3 | 32.4 | 30.9 | 30.6 |
| | | | 60 | 0.0 | 11.2 | 19.4 | 25.6 | 24.0 | 24.0 |
| | | | 90 | 0.0 | 10.1 | 18.2 | 23.9 | 22.9 | 23.1 |
| | | 12 | 30 | 2.0 | 3.1 | 5.5 | 6.6 | 6.3 | 6.0 |
| | | | 60 | 0.0 | 1.8 | 3.3 | 4.1 | 4.1 | 3.0 |
| | | | 90 | 0.0 | 1.7 | 3.3 | 4.0 | 3.9 | 3.0 |
| C7-299 | 299 | 1 | 30 | 5.0 | 14.6 | 22.9 | 27.0 | 28.3 | 29.6 |
| | | | 60 | 0.0 | 9.6 | 18.4 | 22.7 | 23.8 | 24.6 |
| | | | 90 | 0.0 | 8.7 | 17.6 | 21.8 | 23.4 | 24.1 |
| | | 12 | 30 | 1.0 | 3.3 | 4.2 | 4.6 | 5.0 | 5.0 |
| | | | 60 | 0.0 | 2.0 | 3.0 | 3.1 | 3.8 | 3.7 |
| | | | 90 | 0.0 | 1.9 | 2.8 | 3.0 | 3.7 | 3.7 |

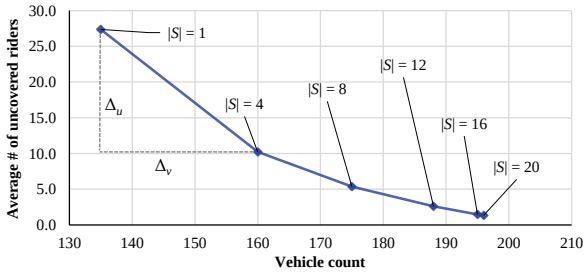



Figure 4.7: Average Uncovered Riders-Vehicle Count Curve of Cluster C2-303 when $f = 1.0$.

well represented by the per-day cost of a parking lot (which in turn is given by its total cost amortized over its useful lifetime). Then suppose a scenario in which $c_u = \$15$ and $c_v = \$8$. The $c_v/c_u$ ratio is $0.53$ which is less than the negative of the slope. Therefore it is beneficial to move from the point with $|\mathcal{S}| = 1$ to that with $|\mathcal{S}| = 4$ as it will result in a marginal cost of $-\$57$. However, suppose an alternate scenario whereby $c_u = c_v = \$12$. The $c_v/c_u$ ratio is now $1.0$ which is larger than the negative of the slope. In this case, it is beneficial to just stay at the point where $|\mathcal{S}| = 1$ as moving to the right neighboring point will result in an increase to the operating cost.

## 4.5 Conclusion

This chapter proposes a two-stage algorithm for generating robust plans for the FCTSP and the RT-CTSP. It addresses a practical setting in which there are uncertainties associated with the schedules of outbound trips by incorporating scenario sampling, a method which assumes the availability of historical data on trip schedules of every commuter to which probability distributions can be fit and sampled to obtain potential scenarios. A model which optimizes the routing plan for a single inbound scenario and multiple sampled outbound scenarios is first solved to select the daily drivers and generate their inbound routes. The outbound routing plan is then obtained either statically for the FCTSP or dynamically for the RT-CTSP by solving another model, one that is defined on just the outbound trips, once their schedules have been confirmed. When applied on a real-world dataset of commute trips, the approach produces plans whose robustness generally increases with the sampled-scenario count. However, the robustness is also accompanied by an increase in vehicle count. Therefore, a method which compares the per-unit price ratio of vehicle increase to uncovered riders is proposed to best evaluate the trade-off between plan robustness and vehicle reduction.

# CHAPTER 5

# The Benefits of Autonomous Vehicles for Community-Based Trip Sharing

## 5.1 Introduction

The notion of community-based trip sharing—leveraging the structure of commuting patterns and urban communities when optimizing trip sharing—was first explored in Chapter 2 to reduce parking pressure and congestion on university and corporate campuses. The study, which was originally motivated by the desire to relieve the parking pressure at the University of Michigan, Ann Arbor, investigated the effects of different driver- and commuter-matching arrangements on trip shareability for a car-pooling or a car-sharing platform. Trip shareability was loosely defined as the ability to aggregate as many trips as possible to reduce the number of vehicles required to serve them. The evaluation of several different optimization models revealed that commuter-matching flexibility, i.e., a willingness to be matched with different drivers and passengers daily, is key for an effective trip-sharing platform.

This early work was extended in Chapter 3 where the CTSP was formalized. The CTSP seeks a car-pool routing plan for a set of commute trips that minimizes a lexicographic objective. The primary objective is to minimize the number of vehicles to cover all the trips, while the secondary objective is to minimize the total travel distance. Every commute trip consists of a pair of trip requests, one to the workplace (inbound) and another back home (outbound), each with specific pickup and drop-off locations as well as time windows specifying allowable service times at each location. Routes of the CTSP must serve each request exactly once and ensure that a vehicle capacity and trip specific ride-duration limits are not exceeded. The ride-duration constraint guarantees a level of QoS for the riders. Finally, as the routing plan selects drivers from the set of commuters, it must ensure that the set of drivers selected for the inbound trips is identical to that for the outbound trips.

The CTSP is thus a VRP with time-window, capacity, pairing, precedence, ride-duration, and driver constraints. Chapter 3 considered two exact algorithms to solve the CTSP and applied them on the real-world dataset from the city of Ann Arbor, Michigan. The case study shows that community-based car pooling can decrease daily vehicle usage by up to 57%. These results highlighted the significant potential in vehicle reduction of community-based trip sharing. However, the vehicles in the CTSP routing plans are still mostly idle, as they perform a single inbound and outbound route a day. Moreover, the constraint that a route starts at the driver origin and ends at the driver destination limits the potential for ride sharing. These limitations make the adoption of AVs for the CTSP particularly appealing, as the absence of drivers would directly address these key shortcomings and could potentially lead to further reductions in fleet size, higher vehicle utilization, and increased ride sharing.

The goal of this chapter is to examine the potential benefits of using AVs for performing the same trips, quantifying the reduction in fleet size and the miles traveled. It studies the CTSPAV which is similar to the CTSP but uses a fleet of AVs that depart from and return to a designated depot to serve all the commute trips. The CTSPAV is very similar to, and is a specialization of, the DARP (Cordeau 2006). The main difference is that trip requests in the CTSPAV come in pairs, one to the workplace (typically in the morning) and another to return back home (typically in the evening). This feature makes it possible to adopt solution techniques that are computationally attractive.

Our proposed solution approach involves chaining mini routes, i.e., short routes with distinct pickup, transit, and drop-off phases, to form longer AV routes. It is especially suited for problem scenarios involving commuters traveling to a common/centralized location, e.g., the commute trips of employees of a university or a corporate campus, or those involving commuters living in a common/centralized location, e.g., the commute trips originating from an apartment complex or a residential neighborhood. In other words, the approach is aimed at problems whose commuting origins and destinations exhibit a hub-and-spoke spatial structure, whereby either the origins or the destinations are concentrated in a common location, as it attempts to leverage the structure to more efficiently solve the problem. Generally speaking, the spatial structure separates the trip origins and destinations into two disjoint "islands". This separation naturally decomposes a long AV route, which travels back and forth between the two islands throughout a day, into a sequence of shorter mini routes, and this structure is then exploited by our proposed algorithm. By contrast, the DARP makes no assumptions with regards to the spatial structure of its trip locations, therefore making it a more general version of the problem. Trips from the Ann Arbor commute-trip dataset fit the profile described, making it a prime candidate for eval-

uating the efficacy of the proposed approach.

The main contributions of this work can be summarized as follows:

1. The work formalizes the CTSPAV that seeks an optimal set of routes for a fleet of AVs for serving a set of commute trips subject to the passenger-related constraints of the original CTSP.
2. The work proposes a column-generation procedure to find a high-quality solution to the problem. The procedure uses a pricing problem to generate mini routes that are then assembled in a master problem. Each mini route serves the inbound (resp. outbound) trips for a number of riders, satisfying the time-window and the ride-duration constraints of the trips as well as the capacity constraints of the vehicles. This approach is a departure from classical column-generation procedures commonly adopted for VRPs and the DARP, whereby the pricing problem searches for complete routes departing from and returning to a depot, while the master problem solves a set-covering/partitioning problem that ensures every customer is served.
3. The work shows that the proposed algorithm outperforms a state-of-the-art DARP algorithm based on the classical column-generation approach for the CTSPAV.
4. The work applies the proposed algorithm on the large-scale, real-world dataset of commute trips from the city of Ann Arbor, Michigan. The experimental results show that the algorithm is capable of reducing daily vehicle usage by 92%, improving upon the results of the original CTSP by 34%, while also reducing daily vehicle miles traveled by approximately 30%.

Overall, the results demonstrate the significant potential of AVs for serving the commuting needs of a community whose members work at a common location. The work also includes a coarse cost analysis that highlights that fleet sizing is the correct metric to optimize when the goal consists of maximizing the profitability of the service.

The rest of this chapter is organized as follows. Section 5.2 specifies the CTSPAV and presents a MIP model that formalizes the CTSPAV. Section 5.3 describes the column-generation procedure. Section 5.4 sketches a DARP-based procedure that is used for comparison purposes. Section 5.5 presents detailed computational results and examines the performance of the proposed approach for servicing the commuting needs of the community under study. Finally, Section 5.6 provides some concluding remarks.

## 5.2 The Commute Trip-Sharing Problem for Autonomous Vehicles

This section specifies the CTSPAV that seeks a set of AV routes of minimal cost to serve each inbound and outbound trip of a set of commuters $\mathcal{C}$ exactly once. Let $n = |\,\mathcal{C}\,|$ denote the total number of commuters, $\mathcal{P}^+ = \{1, \ldots, n\}$ and $\mathcal{D}^+ = \{n+1, \ldots, 2n\}$ denote the sets of all pickup and drop-off nodes of inbound trips respectively, and $\mathcal{P}^- = \{2n+1, \ldots, 3n\}$ and $\mathcal{D}^- = \{3n+1, \ldots, 4n\}$ denote the sets of all pickup and drop-off nodes of outbound trips respectively. Let $\mathcal{P} = \mathcal{P}^+ \cup \mathcal{P}^-$ and $\mathcal{D} = \mathcal{D}^+ \cup \mathcal{D}^-$. The nodes have been defined such that the inbound pickup, inbound drop-off, outbound pickup, and outbound drop-off locations of commuter $i$ are represented by nodes $i$, $n+i$, $2n+i$, and $3n+i$ respectively, and $n+i$ gives the corresponding drop-off node of pickup node $i \in \mathcal{P}$.

Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ denote a directed graph with the node set $\mathcal{N} = \mathcal{P} \cup \mathcal{D} \cup \{v_s, v_t\}$ containing all pickup and drop-off nodes together with a source and a sink node representing a designated depot. A ride-duration limit $L_i$ is associated with each node $i \in \mathcal{P}$. A time window $[a_i, b_i]$ and a service duration $\zeta_i$ are also associated with each node $i \in \mathcal{P} \cup \mathcal{D}$. There are no time-window constraints for the start and end times of any AV route, as it is assumed that the AVs may start and end their routes at any time of the day. In a first approximation, the edge set $\mathcal{A} = \{(i,j) : i, j \in \mathcal{N}, i \neq j\}$ consists of all possible edges. A travel time $\tau_{(i,j)}$, a distance $\varsigma_{(i,j)}$, and a cost $c_{(i,j)}$ are associated with each edge $(i,j) \in \mathcal{A}$. The sets of all outgoing and incoming edges of node $i$ are denoted by $\delta^+(i)$ and $\delta^-(i)$ respectively. By definition of AV routes, the following precedence constraints apply to the set of nodes:

$$i \prec n+i \prec 2n+i \prec 3n+i \qquad \forall i \in \mathcal{P}^+ \tag{5.1}$$

where $i \prec j$ denotes the precedence relation between nodes $i$ and $j$, i.e., the constraint indicating that $i$ must be visited before $j$ on an AV route.

This work considers two distinct optimization objectives: (1) a lexicographic objective that first minimizes the number of vehicles and then their total travel distance, and (2) a single objective that only minimizes the total travel distance.

### 5.2.1 A MIP Model for the CTSPAV

This section presents a MIP model for the CTSPAV. The MIP formalizes the CTSPAV and is the foundation of the column-generation procedure presented in the next section. It is defined in terms of the set $\Omega$ of all feasible mini routes and the graph $\mathcal{G}$.

The MIP, also referred to as the master problem ($\text{MP}_{\text{CTSPAV}}$), is defined by (5.2)–(5.12).

It uses two sets of binary variables: variable $X_r$ indicates whether mini route $r \in \Omega$ is selected and variable $Y_e$ indicates whether edge $e \in \mathcal{A}$ is used in the optimal routing plan. It also uses a continuous variable $T_i$ to represent the start of service time at node $i \in \mathcal{P} \cup \mathcal{D}$. The model minimizes the total cost of all selected edges. Constraints (5.3) enforce coverage of each trip by exactly one mini route, while constraints (5.4) ensure that edges belonging to selected mini routes are selected. Constraints (5.5) and (5.6) conserve flow through each pickup and drop-off node while ensuring each is visited exactly once. Constraints (5.7) and (5.8) enforce compatibility of service start times and travel times along selected edges by utilizing large constants for $M_{(i,j)}$ and $\bar{M}_{(i,j)}$. Constraints (5.9) describe the ride-duration limit for each trip, while constraints (5.10) are time-window constraints for all pickup and drop-off nodes.

$$\min \quad \sum_{e \in \mathcal{A}} c_e Y_e \tag{5.2}$$

$$\text{s.t.} \quad \sum_{r \in \Omega : i \in r} X_r = 1 \qquad \forall i \in \mathcal{P} \tag{5.3}$$

$$\sum_{r \in \Omega : e \in r} X_r - Y_e \leq 0 \qquad \forall e \in \mathcal{A} \setminus \{\delta^+(v_s) \cup \delta^-(v_t)\} \tag{5.4}$$

$$\sum_{e \in \delta^+(i)} Y_e = 1 \qquad \forall i \in \mathcal{P} \cup \mathcal{D} \tag{5.5}$$

$$\sum_{e \in \delta^-(i)} Y_e = 1 \qquad \forall i \in \mathcal{P} \cup \mathcal{D} \tag{5.6}$$

$$T_i + \zeta_i + \tau_{(i,j)} \leq T_j + M_{(i,j)}(1 - Y_{(i,j)}) \qquad \forall i, j \in \mathcal{P} \cup \mathcal{D} \tag{5.7}$$

$$T_i + \zeta_i + \tau_{(i,j)} \geq T_j - \bar{M}_{(i,j)}(1 - Y_{(i,j)}) \qquad \forall i \in \mathcal{P} \cup \mathcal{D}, \forall j \in \mathcal{D} \tag{5.8}$$

$$T_{i+n} - (T_i + \zeta_i) \leq L_i \qquad \forall i \in \mathcal{P} \tag{5.9}$$

$$a_i \leq T_i \leq b_i \qquad \forall i \in \mathcal{P} \cup \mathcal{D} \tag{5.10}$$

$$X_r \in \{0, 1\} \qquad \forall r \in \Omega \tag{5.11}$$

$$Y_e \in \{0, 1\} \qquad \forall e \in \mathcal{A} \tag{5.12}$$

The lexicographic objective is accomplished using a blended approach that appropriately weights the sub-objectives: It assigns an identical, large fixed cost to each AV route and a variable cost to each that is proportional to its total distance. Let $\mathcal{R}$ denote the set of all feasible AV routes. The edge costs are then defined as follows:

$$c_e = \begin{cases} \varsigma_e + 100 \cdot \hat{\varsigma}_{\text{max}} & \forall e \in \delta^+(v_s) \\ \varsigma_e & \text{otherwise} \end{cases} \tag{5.13}$$

where $\hat{\varsigma}_{\text{max}}$ is a constant equal to the length (total distance) of the longest AV route, i.e.:

$$\hat{\varsigma}_{\text{max}} = \max_{\rho \in \mathcal{R}} \sum_{(i,j) \in \rho} \varsigma_{(i,j)} \tag{5.14}$$

The fixed cost, $100 \cdot \hat{\varsigma}_{\text{max}}$, that is significantly larger than the total length of any AV route and that is also identical for every edge $e \in \delta^+(v_s)$, drives the model to first minimize the total flow emanating from the depot, $v_s$. This flow is identical to the total number of AV routes, and thus the number of vehicles used in the solution. The variable cost $\varsigma_e$ of every edge $e$ then drives the model to minimize the total distance of all selected edges. The edge costs defined in (5.13) therefore accomplish the desired lexicographic ordering of the objective, which first minimizes the number of vehicles used in the solution and then their total travel distance.

Conversely, when the objective is to just minimize the total distance, the edge costs are simply defined by the edge distance, i.e.,

$$c_e = \varsigma_e \qquad \forall e \in \mathcal{A} \tag{5.15}$$

The $\text{MP}_{\text{CTSPAV}}$ model can be seen as a scheduling problem that selects and assembles feasible mini routes to form longer, feasible AV routes that minimize the total cost. The optimal AV routes are obtained by constructing paths beginning at $v_s$ and ending at $v_t$. The start and end times (at the depot) of these routes are then obtained by applying equations (1.7) and (1.9) respectively.

## 5.3 A Column-Generation Procedure for the CTSPAV

This section presents a column-generation approach to find high-quality solutions to the CTSPAV, referred to as the CTSPAV procedure. The column-generation approach builds on $\text{MP}_{\text{CTSPAV}}$ but addresses its main computational difficulty: the fact that $\text{MP}_{\text{CTSPAV}}$ assumes that all mini routes have been pre-computed. The column-generation approach implements an iterative process that considers, in each iteration, a subset $\Omega' \subseteq \Omega$ of feasible mini routes and solves a restricted master problem, denoted by $\text{RMP}_{\text{CTSPAV}}$, that is defined as the linear relaxation of $\text{MP}_{\text{CTSPAV}}$ over $\Omega'$. Using the dual information from $\text{RMP}_{\text{CTSPAV}}$, the column-generation algorithm then searches for feasible mini routes with negative reduced costs by solving a pricing subproblem ($\text{PSP}_{\text{CTSPAV}}$). If such mini routes exist, they are added to $\Omega'$. Each iteration thus defines a new restricted master problem over a larger subset of feasible mini routes. The solving of restricted master problems and pricing sub-

problems is repeated until the pricing subproblem cannot find any feasible mini routes with a negative reduced cost. Upon completion, the optimal objective value of $\text{RMP}_{\text{CTSPAV}}$ *converges* to $z^*$, the optimal objective value of the linear relaxation of $\text{MP}_{\text{CTSPAV}}$. Whenever the solution of $\text{RMP}_{\text{CTSPAV}}$ is integral at convergence, it is also optimal for $\text{MP}_{\text{CTSPAV}}$. Otherwise, the column-generation approach solves the final restricted master problem as a MIP to obtain an integer solution. The objective value of the MIP provides an upper bound to the optimal solution, while the objective value $z^*$ of its linear relaxation provides a lower bound. Together, they are used to compute an optimality gap for the integer solution.

The approach adopted in this procedure, which has its pricing subproblem search for feasible mini routes which are then chained together in the master problem to form longer, feasible AV routes that depart from and return to the depot, is unlike the classical column-generation approach adopted in most literature on the VRPTW (e.g. Desrosiers et al. (1984), Desrochers et al. (1992)), PDPTW (e.g. Dumas et al. (1991), Ropke and Cordeau (2009)), or DARP (e.g. Gschwind and Irnich (2015)). The classical approach is an application of the Dantzig-Wolfe decomposition on an edge-flow formulation of the problem: It produces a set-partitioning/covering master problem that just selects feasible routes from a set to ensure every customer is served in the solution. Its pricing problem is then solely responsible for searching for *complete* feasible routes that originate from and return to the depot (i.e., routes that would correspond to the AV routes of the CTSPAV). By contrast, our approach shifts part of the burden of constructing the AV routes, which are anticipated to be very long, to the master problem instead of completely relegating the task to the pricing subproblem.

### 5.3.1 The Pricing Subproblem

The $\text{PSP}_{\text{CTSPAV}}$ identifies feasible mini routes with negative reduced costs. Let $\{\pi_i : i \in \mathcal{P}\}$ and $\{\mu_e : e \in \mathcal{A} \setminus \{\delta^+(v_s) \cup \delta^-(v_t)\}\}$ denote the dual values associated with constraints (5.3) and (5.4) at optimality of $\text{RMP}_{\text{CTSPAV}}$ respectively. The reduced cost of a mini route $r$ is then given by:

$$\bar{c}_r = -\sum_{i \in r : i \in \mathcal{P}} \pi_i - \sum_{e \in r} \mu_e \tag{5.16}$$

The column-generation approach attempts to generate multiple feasible mini routes during each iteration, one for each pickup node. More precisely, $\text{PSP}_{\text{CTSPAV}}$ considers each node $i \in \mathcal{P}^+ \cup \mathcal{P}^-$ as the starting point of a mini route $r_i$. For each $i \in \mathcal{P}^+ \cup \mathcal{P}^-$, it searches for the mini route $r_i$ with minimal reduced cost and selects those with negative reduced costs to augment $\Omega'$. It accomplishes this by first constructing $2n$ graphs, $\mathcal{G}_i^+ (i \in \mathcal{P}^+)$ and $\mathcal{G}_i^- (i \in \mathcal{P}^-)$. It then searches for the least-cost path from $i$ to a designated sink node that

satisfies all mini-route feasibility constraints from each graph. The complete details of this procedure are given in Appendix B.1.

### 5.3.2 Practical Implementation Considerations

This subsection reviews a number of important implementation techniques for the CTSPAV column-generation procedure.

**Filtering of Graph $\mathcal{G}$** Many edges in $\mathcal{G}$ do not belong to any feasible AV route and can be removed from $\mathcal{A}$. The following sets of infeasible edges are obtained by pre-processing time-window, pairing, precedence, and ride-duration limit constraints on $\mathcal{A}$ using a combination of rules proposed by Dumas et al. (1991) and Cordeau (2006):

(a) Direct trips to and from the depot:
- $\{(v_s, v_t), (v_t, v_s)\}$
- $\{(i, v_s), (i, v_t), (v_t, i) : i \in \mathcal{P}\}$
- $\{(v_s, i), (i, v_s), (v_t, i) : i \in \mathcal{D}\}$

(b) Pairing and precedence of pickup and drop-off nodes of inbound and outbound trips of each commuter (constraints (5.1)): $\{(i, 2n+i), (i, 3n+i), (n+i, i), (n+i, 3n+i), (2n+i, i), (2n+i, n+i), (3n+i, i), (3n+i, n+i), (3n+i, 2n+i) : i \in \mathcal{P}^+\}$

(c) Time windows along each edge: $\{(i,j) : (i,j) \in \mathcal{A} \setminus \{\delta^+(v_s) \cup \delta^-(v_t)\} \wedge a_i + \zeta_i + \tau_{(i,j)} > b_j\}$

(d) Ride-duration limit of each commuter: $\{(i,j), (j, n+i) : i \in \mathcal{P} \wedge j \in \mathcal{P} \cup \mathcal{D} \wedge i \neq j \wedge \tau_{(i,j)} + \zeta_j + \tau_{(j,n+i)} > L_i\}$

(e) Time windows and ride-duration limits of pairs of trips:
- $\{(i, n+j) : i, j \in \mathcal{P} \wedge i \neq j \wedge \neg feasible(j \to i \to n+j \to n+i)\}$
- $\{(n+i, j) : i, j \in \mathcal{P} \wedge i \neq j \wedge \neg feasible(i \to n+i \to j \to n+j)\}$
- $\{(i,j) : i, j \in \mathcal{P} \wedge i \neq j \wedge \neg feasible(i \to j \to n+i \to n+j) \wedge \neg feasible(i \to j \to n+j \to n+i)\}$
- $\{(n+i, n+j) : i, j \in \mathcal{P} \wedge i \neq j \wedge \neg feasible(i \to j \to n+i \to n+j) \wedge \neg feasible(j \to i \to n+i \to n+j)\}$

Note that the sets of edges in (e) utilize the $feasible$ function described in Section 1.1.3 to determine if a partial route satisfies the time-window and the ride-duration limit constraints. For instance, the first condition indicates that edge $(i, n+j)$ is infeasible if route $j \to i \to n+j \to n+i$ is infeasible. Figure 5.1 illustrates an example of graph $\mathcal{G}$ resulting from the removal of the infeasible edges.

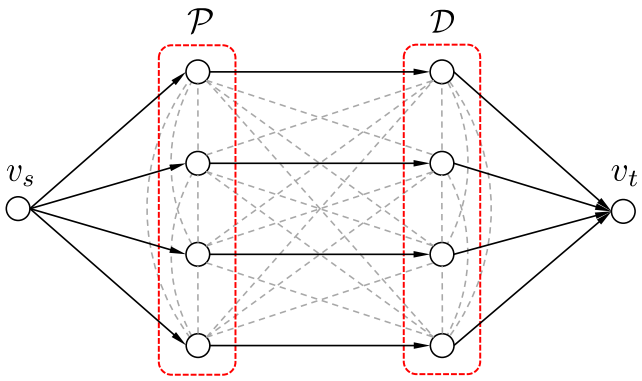



Figure 5.1: Graph $\mathcal{G}$ (Each Dotted Line Represents a Pair of Bidirectional Edges).

**Big-M Constants** The $\text{RMP}_{\text{CTSPAV}}$ utilizes big-$M$ constants in constraints (5.7) and (5.8) to enforce the underlying constraints *only* on selected edges. To ensure that the constants are large enough to accomplish this goal while not being excessively large so as to introduce numerical issues, they are defined as follows:

$$M_{(i,j)} = \max\{0, b_i + \zeta_i + \tau_{(i,j)} - a_j\} \qquad \forall i, j \in \mathcal{P} \cup \mathcal{D} \tag{5.17}$$

$$\bar{M}_{(i,j)} = \max\{0, b_j - a_i - \zeta_i - \tau_{(i,j)}\} \qquad \forall i \in \mathcal{P} \cup \mathcal{D}, \forall j \in \mathcal{D} \tag{5.18}$$

When the lexicographic objective is considered, the edge cost defined in (5.13) uses $\hat{\varsigma}_{\text{max}}$ which denotes the length of the longest AV route. Since enumeration of all feasible AV routes in $\mathcal{R}$ is impractical, a conservative overestimate is used for $\hat{\varsigma}_{\text{max}}$ to accomplish the lexicographic ordering of the sub-objectives.

**Lower Bound** Column-generation procedures are known to have a tailing-off effect, whereby the rate-of-change of the $\text{RMP}_{\text{CTSPAV}}$ objective value $z_{\text{RMP}_{\text{CTSPAV}}}$ progressively decreases as $z_{\text{RMP}_{\text{CTSPAV}}}$ approaches $z^*$ (Lübbecke and Desrosiers 2005). To mitigate this effect, a dual lower bound to $z^*$, $z_{\text{LB}}$, is defined using the generalized version of the Lasdon bound (Lasdon 1970), i.e.,

$$z_{\text{LB}} = z_{\text{RMP}_{\text{CTSPAV}}} + \kappa \bar{c}_r^* \tag{5.19}$$

where $\kappa$ is an upper bound to the number of selected mini routes in $\text{MP}_{\text{CTSPAV}}$, $\kappa \geq \sum_{r \in \Omega} X_r$, and $\bar{c}_r^*$ is the smallest mini-route reduced cost discovered from $\text{PSP}_{\text{CTSPAV}}$. For this problem, it is sufficient to take $\kappa = 2n$. Since the edge costs are all integral, the optimal objective value of $\text{MP}_{\text{CTSPAV}}$ must also be integral, and therefore the column-generation iterations can be terminated when $\lceil z_{\text{RMP}_{\text{CTSPAV}}} \rceil - z_{\text{LB}} < 1$.

**Solving the Subproblem** The label-setting algorithm of Gschwind and Irnich (2015) that is used to solve $\text{PSP}_{\text{CTSPAV}}$ produces an intermediate set of non-dominated, feasible mini routes, $\hat{\Omega}_i$ for each graph $\mathcal{G}_i^+ (i \in \mathcal{P}^+)$ and $\mathcal{G}_i^- (i \in \mathcal{P}^-)$. Instead of considering only the least-cost route from $\hat{\Omega}_i$, all routes from $\hat{\Omega}_i$ with negative reduced costs are selected and introduced into $\Omega'$ to further accelerate the column-generation convergence. Moreover, since the mini-route search procedure on all graphs are independent, they are solved concurrently in our implementation. Finally, $\Omega'$ is initialized with the set of all direct-trip routes, i.e., it is initialized with $\{i \rightarrow n + i : i \in \mathcal{P}\}$.

## 5.4 The DARP Column-Generation Procedure

The CTSPAV can be viewed as a specialization of the DARP: It can be converted into a DARP by simply setting the time window of each vehicle at the depot to $\pm\infty$. This section describes a column-generation procedure, referred to as the DARP procedure, that is derived from the algorithm for solving the DARP by Gschwind and Irnich (2015). It is the algorithm to which the CTSPAV procedure is compared in the computational results section. At a high level, the DARP procedure is similar to the CTSPAV procedure as they both use column generation. However, the DARP procedure fundamentally differs from the CTSPAV procedure in that it adopts the classical column-generation approach. More specifically, the DARP procedure uses a set-covering restricted master problem $\text{RMP}_{\text{DARP}}$ that only selects AV routes from a set $\mathcal{R}'$ to ensure every trip is covered in the solution. Columns of $\text{RMP}_{\text{DARP}}$ represent AV routes whereas those of $\text{RMP}_{\text{CTSPAV}}$ represent mini routes. The procedure also uses a pricing subproblem $\text{PSP}_{\text{DARP}}$ that searches for feasible AV routes to augment $\mathcal{R}'$. Upon convergence of the column-generation process, the $\text{RMP}_{\text{DARP}}$ is solved as a MIP to obtain an integer solution.[1]

### 5.4.1 The Master Problem

The master problem $\text{MP}_{\text{DARP}}$ is a set-covering formulation that seeks the optimal routing plan for the CTSPAV. It is defined on the set of all feasible AV routes $\mathcal{R}$ and uses a binary variable $X_\rho$ that indicates whether route $\rho \in \mathcal{R}$ is used in the plan. The model is given by

[1]The branch-and-price approach proposed by Gschwind and Irnich (2015) is not considered because it is found to be too expensive for the problem instances used in this work. Even the root node of the branch-and-price tree cannot be solved within the allocated time budget for the real instances considered in this work.

(5.20)–(5.22).

$$\min \quad \sum_{\rho \in \mathcal{R}} c_\rho X_\rho \tag{5.20}$$

$$\text{s.t.} \quad \sum_{\rho \in \mathcal{R}} a_{i,\rho} X_\rho \geq 1 \qquad \forall i \in \mathcal{P} \tag{5.21}$$

$$X_\rho \in \{0, 1\} \qquad \forall \rho \in \mathcal{R} \tag{5.22}$$

The objective function (5.20) minimizes the total cost of the selected routes. Constant $a_{i,\rho}$ in constraints (5.21) represents the number of times node $i$ is visited by route $\rho$. These constraints ensure that each pickup node is covered in the optimal plan. A set-covering formulation is preferred in this work as it was discovered to produce stronger integer solutions than a set-partitioning formulation based on preliminary experimental evaluations.

To find a routing plan that minimizes vehicle count, the cost $c_\rho$ of each route is set to 1. On the other hand, to find a plan that minimizes the total travel distance, the cost $c_\rho$ is set to the total distance of $\rho$, i.e., $c_\rho = \sum_{(i,j) \in \rho} \varsigma_{(i,j)}$. Finally, to implement a lexicographic objective that first minimizes the vehicle count and then their total distance, the model is solved twice. The model is first solved to produce the optimal vehicle count $\chi^*_{\text{MIP}}$. The constraint

$$\sum_{\rho \in \mathcal{R}} X_\rho = \chi^*_{\text{MIP}} \tag{5.23}$$

is then introduced to the model to fix the vehicle count to its optimal value, after which the model is solved again to optimize the secondary objective.

While a blended approach similar to that used in the CTSPAV procedure could have also been used here to implement the lexicographic objective, initial experimental evaluations revealed that the greater complexity of $\text{PSP}_{\text{DARP}}$, which is significantly more expensive than $\text{PSP}_{\text{CTSPAV}}$, combined with the use of the Lasdon bound (5.19) results in a column-generation phase that converges significantly slower. The proposed multi-objective approach, which first just minimizes the vehicle count and therefore uses identical costs for the routes of $\text{RMP}_{\text{DARP}}$ (unlike route costs for the blended approach), permits the use of the dual bound proposed by Farley (1990) in the column-generation phase which is stronger than the Lasdon bound in this setting. This stronger dual bound consequently allows the column-generation termination criterion to be satisfied earlier, thus resulting in a faster converging column-generation phase for the primary objective. And while a similar approach could have also been used for the CTSPAV procedure, the less expensive nature of $\text{PSP}_{\text{CTSPAV}}$ makes a strong dual bound less critical for its column-generation phase which already converges quickly. In the end, the blended and the multi-objective approaches

are different yet valid alternatives for implementing the lexicographic objective. The latter, which applies the lexicographic ordering directly, is preferred for the DARP procedure simply because it allows the column-generation phase for the primary objective to converge quicker in practice and is seen as a necessity to counteract the increased complexity of its pricing subproblem.

### 5.4.2 The Pricing Subproblem

The $\text{PSP}_{\text{DARP}}$ searches for AV routes with negative reduced costs. Let $\{\alpha_i : i \in \mathcal{P}\}$ denote the set of optimal duals of constraints (5.21) and $\beta$ be that of constraint (5.23). When $\text{RMP}_{\text{DARP}}$ has the vehicle-count minimization objective, the reduced cost of route $\rho$ is given by:

$$\bar{c}_\rho = 1 - \sum_{i \in \mathcal{P}} a_{i,\rho} \alpha_i \tag{5.24}$$

When the distance-minimization objective is applied, the reduced cost of $\rho$ is given by:

$$\bar{c}_\rho = \sum_{(i,j) \in \rho} \varsigma_{(i,j)} - \sum_{i \in \mathcal{P}} a_{i,\rho} \alpha_i \tag{5.25}$$

Finally, when constraint (5.23) is also present in $\text{RMP}_{\text{DARP}}$ with the distance-minimization objective, the reduced cost of $\rho$ is given by:

$$\bar{c}_\rho = \sum_{(i,j) \in \rho} \varsigma_{(i,j)} - \sum_{i \in \mathcal{P}} a_{i,\rho} \alpha_i - \beta \tag{5.26}$$

The pricing subproblem searches for routes with negative reduced costs using a graph $\mathcal{G}$ that is identical to the one described in Section 5.2. A reduced cost $\bar{c}_{(i,j)}$ is assigned to each edge $(i,j) \in \mathcal{A}$, and it is defined according to the objective function used. For the vehicle-count minimization objective, $\bar{c}_{(i,j)}$ is given by:

$$\bar{c}_{(i,j)} = \begin{cases} 1 & \forall (i,j) \in \delta^+(v_s) \\ -\alpha_i & \forall i \in \mathcal{P}, \forall j \in \mathcal{N} \\ 0 & \forall i \in \mathcal{D}, \forall j \in \mathcal{N} \end{cases} \tag{5.27}$$

When the distance-minimization objective is used, $\bar{c}_{(i,j)}$ is given by:

$$\bar{c}_{(i,j)} = \begin{cases} \varsigma_{(i,j)} - \alpha_i & \forall i \in \mathcal{P}, \forall j \in \mathcal{N} \\ \varsigma_{(i,j)} & \forall i \in \mathcal{D} \cup \{v_s\}, \forall j \in \mathcal{N} \end{cases} \tag{5.28}$$

The edge reduced costs are defined so that the total cost of any path in $\mathcal{G}$ from $v_s$ to $v_t$ is equivalent to the reduced cost of the path defined in (5.24)–(5.26). In the presence of constraint (5.23) for the distance-minimization objective, (5.28) is still used to define the edge reduced costs and $-\beta$ is just added to the final path cost to obtain the reduced cost defined in (5.26).

Once $\mathcal{G}$ has been set up with the proper edge reduced costs, $\text{PSP}_{\text{DARP}}$ just searches for the least-cost path from $v_s$ to $v_t$ that satisfies the time-window, vehicle-capacity, pairing, precedence, and ride-duration limit constraints. This least-cost path is then added to $\mathcal{R}'$ if the cost is negative. The problem is an SPPRC and is solved using the label-setting dynamic program proposed by Gschwind and Irnich (2015) which utilizes resource constraints to enforce the route-feasibility constraints.

In the presence of negative-cost cycles in $\mathcal{G}$, the label-setting algorithm may produce non-elementary paths. Whereas an additional resource may be introduced to the label-setting algorithm to guarantee path elementarity, the SPPRC then becomes an ESPPRC which is extremely hard to solve. Therefore, our implementation adopts a strategy from Ropke and Cordeau (2006) and Gschwind and Irnich (2015) which simply relaxes this elementarity requirement. While doing so theoretically causes $\text{RMP}_{\text{DARP}}$ to converge to a weaker primal lower bound as it now admits a larger set of routes $\mathcal{R}'' \supseteq \mathcal{R}'$, both Ropke and Cordeau (2009) and Gschwind and Irnich (2015) have found that the resulting lower bound is only slightly weaker in practice.

### 5.4.3 Implementation Strategies

Similar to the CTSPAV procedure, a dual lower bound is maintained during the column-generation procedure to mitigate its tailing-off effect. When the vehicle-count minimization objective is active, the lower bound proposed by Farley (1990) is adopted since the cost of each route is identical. It is given by:

$$z'_{\text{LB}} = \frac{z_{\text{RMP}_{\text{DARP}}}}{1 - \bar{c}^*_\rho} \tag{5.29}$$

where $z_{\text{RMP}_{\text{DARP}}}$ is the objective value of $\text{RMP}_{\text{DARP}}$ at the end of each iteration and $\bar{c}^*_\rho$ is the smallest route reduced cost discovered by $\text{PSP}_{\text{DARP}}$. The column generation is then terminated when $\lceil z_{\text{RMP}_{\text{DARP}}} \rceil - z'_{\text{LB}} < 1$. To accomplish the lexicographic objective during the column generation, constraint (5.23) is introduced to $\text{RMP}_{\text{DARP}}$ after the primary objective has converged with its right-hand side set to $\lceil z_{\text{RMP}_{\text{DARP}}} \rceil$. The objective function is then switched to distance minimization after which the column generation is resumed. For the

distance-minimization objective, the generalized Lasdon bound defined in (5.19) is used as the dual lower bound, and the column generation is terminated when $\lceil z_{\text{RMP}_{\text{DARP}}} \rceil - z_{\text{LB}} < 1$.

Our implementation also incorporates the interior-point, dual-stabilization method proposed by Rousseau et al. (2007) to accelerate the column-generation convergence. Furthermore, all non-dominated routes with negative reduced costs resulting from the label-setting algorithm in $\text{PSP}_{\text{DARP}}$ are added to $\mathcal{R}'$. The column-generation phase is seeded with routes $\{i \to n+i \to 2n+i \to 3n+i : i \in \mathcal{P}^+\}$. Since this set of routes represents a feasible solution to the problem, it guarantees the existence of a feasible integer solution to the MIP for both the distance-minimization and the primary lexicographic objective. Recall that for the secondary lexicographic objective, the right-hand side of the introduced constraint (5.23) is set to $\chi^*_{\text{MIP}}$ which represents the objective value of the optimal integer solution for the primary objective. The existence of this feasible integer solution guarantees that one also exists for the MIP with the secondary objective.

Non-elementary routes produced in the column-generation phase are removed prior to solving $\text{RMP}_{\text{DARP}}$ as a MIP. Repeated nodes are identified from each non-elementary route and only the first instance of each repeated node is preserved in the route (subsequent instances are removed). The resulting route is feasible as the non-elementary version already satisfies the time-window, vehicle-capacity, pairing, precedence, and ride-duration limit constraints. Similarly, since a set-covering formulation is used, a node may be visited by multiple routes in the integer solution. This is fixed by simply preserving the node in an arbitrarily selected route and removing it from the others. Since the travel distances satisfy the triangle inequality, this step only shortens the affected routes and hence improves the total travel distance of the solution while maintaining its vehicle count.

## 5.5 Case Study and Experimental Results

This section evaluates the potential benefits of AVs on a real case study. It also reports a variety of experimental results on the efficiency of the optimization algorithms.

### 5.5.1 The Dataset and Construction of Problem Instances

The performance of the CTSPAV and the DARP procedures are evaluated on problem instances derived from the Ann Arbor commute-trip dataset. The results in this section focus on the busiest days (Monday–Thursday) of the busiest week of the month (week 2). For additional perspectives, the trips are partitioned into two sets: the approximately 2,200 daily commute trips made by the commuters living within Ann Arbor city limits (the region

bounded by highways US-23, M-14, and I-94), and the remainder made by the commuters living outside the region. The rationale behind this split is to distinguish between the results of the trips of commuters living nearer to the downtown area, where the parking structures are located, and those of commuters living further away. The trips from each set are then further partitioned into smaller problem instances using the spatial clustering algorithm described in Section 2.2.1 which groups at most $N$ commuters together based on the spatial proximity of their home locations. The clusters are thought of as "neighborhoods" within which trip sharing is done exclusively. This notion of breaking down a problem into smaller instances is in line with the conclusion by Agatz et al. (2012), that effective decomposition techniques are necessary for the computational feasibility of large-scale problems.

Several assumptions are made to characterize the nature of the requests submitted to the trip-sharing platform. First, each rider $i$, when requesting a trip, specifies a desired arrival time $at_i^+$ to the inbound trip destination and a desired departure time $dt_i^-$ from the outbound trip origin. This assumption is consistent with those in the literature on the DARP, e.g., Jaw et al. (1986), Cordeau and Laporte (2003b), and Cordeau (2006). Secondly, each rider tolerates a maximum shift of $\pm\Delta$ to the desired times. Therefore, if the arrival and the departure times at the parking structures from the dataset are considered as the desired times, then a delivery-time upper bound of $b_i = at_i^+ + \Delta$ can be associated with each $i \in \mathcal{D}^+$ and a pickup time window of $[a_i, b_i] = [dt_i^- - \Delta, dt_i^- + \Delta]$ can be associated with each $i \in \mathcal{P}^-$. Consequently, a departure time window of $[a_i, b_i] = [b_{n+i} - \zeta_i - L_i - 2\Delta, b_{n+i} - \zeta_i - L_i]$ is associated with each $i \in \mathcal{P}^+$ and a delivery-time upper bound of $b_i = b_{i-n} + \zeta_{i-n} + L_{i-n}$ is associated with each $i \in \mathcal{D}^-$. Finally, each rider $i$ will tolerate at most an $R\%$ extension to her direct-trip ride duration, i.e., $L_i = (1 + R)\tau_{(i,n+i)}$ for each $i \in \mathcal{P}$. Hunsaker and Savelsbergh (2002) also used a similar assumption. This use of a single multiplication factor for modeling the maximum tolerable ride duration (versus a more fine-grained approach) is seen as a practical necessity to cater to the massive volume of trips considered in our case study, and it is seen as a more realistic alternative to the approach used in other works on the DARP (e.g., Cordeau (2006) used an identical duration for every trip). Results of sensitivity analyses on both $\Delta$ and $R$ are provided in later sections to demonstrate their effect on the final results.

**Depot Configurations** This work explores two hypothetical depot configurations: (1) a central depot configuration in which all neighborhoods are served by vehicles from a single, centralized depot, and (2) a local depot configuration whereby each neighborhood is served by a local depot situated within the neighborhood itself. For the first scenario, the largest parking structure from the dataset considered is arbitrarily designated as the central depot.

In the second scenario, the home address $l_c$ that is nearest to every other location within cluster $\mathcal{P}_c$ is selected as the hypothetical local depot location, i.e.,

$$l_c = \arg\min_{i \in \mathcal{P}_c} \left\{ \sum_{j \in \mathcal{P}_c \setminus \{i\}} \varsigma_{(i,j)} + \varsigma_{(j,i)} \right\}. \tag{5.30}$$

### 5.5.2 Experimental Setup and Parameters

The GPS coordinates of every address considered are geocoded using Geocodio, while the shortest path, travel time, and travel distance between any two locations are estimated using GraphHopper's Directions API that uses OpenStreetMap data. All algorithms are implemented in C++ with parallelization being handled with OpenMP. The label-setting algorithm of Gschwind and Irnich (2015) is implemented using the resource-constrained shortest path framework from the Boost Graph Library (version 1.70.0), while all LPs and MIPs are solved with Gurobi 8.1.1. Every problem instance is solved on a compute cluster, utilizing 12 cores of a 3.0 GHz Intel Xeon Gold 6154 processor and 32 GB of RAM. A total time budget of 2 hours is allocated for each instance; 1 hour for the column-generation phase and another 1 hour for solving the MIP for the CTSPAV procedure. The same total time budget is allocated for the DARP procedure. However, since preliminary evaluations revealed that it requires more time for its column-generation phase and less for solving the MIP, 1.5 hours is allocated for its column-generation phase and 0.5 hours for solving its MIP. All reported results are from the best feasible solution obtained within the time limit.

The experiments consider the use of autonomous cars and hence $K = 4$. Preliminary empirical evaluations found that $N = 100$ for the clustering algorithm produces neighborhoods that are sufficiently large to provide ample opportunities for trip sharing while not producing intractable problem instances. This setting was thus used to generate the problem instances for all experiments. Finally, unless otherwise stated, values of $\Delta = 10$ mins and $R = 50\%$ are used in all experiments. Appendix B.2 summarizes the optimality gaps and the computation times of the procedures. Both the appendix and the next subsection summarizes the results of all problem instances obtained from applying the clustering algorithm on all commute trips from the Wednesday of week 2 (which generated 22 and 68 clusters inside and outside the city respectively).

### 5.5.3 Performance Comparison of the CTSPAV and DARP Procedures

This section presents a comparison of the results of the CTSPAV procedure against those of the DARP, particularly the final objective values of their MIPs together with the corre-

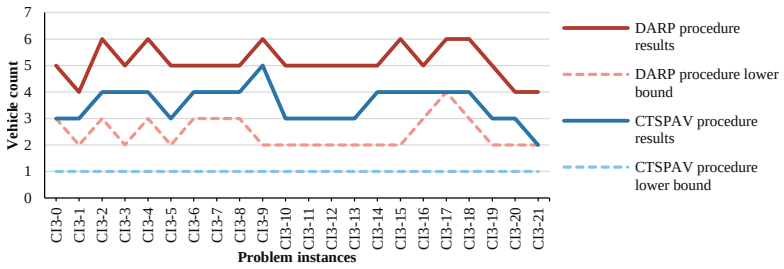



Figure 5.2: Comparison of Vehicle Count Results for Problem Instances Inside City Limits with the Lexicographic Objective and the Central Depot Configuration.

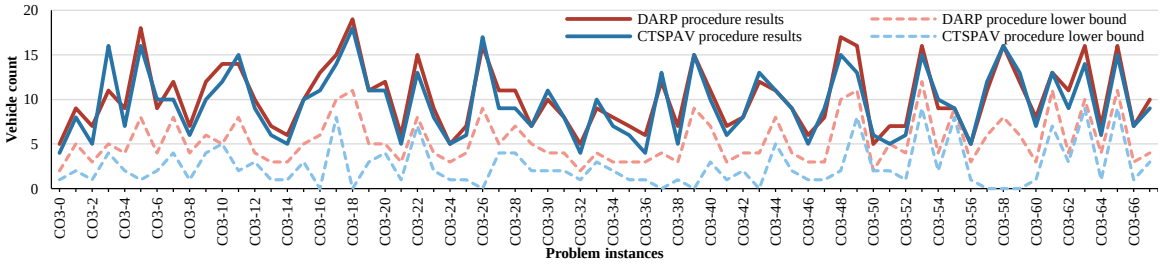



Figure 5.3: Comparison of Vehicle Count Results for Problem Instances Outside City Limits with the Lexicographic Objective and the Central Depot Configuration.

sponding lower bounds of each procedure for every problem instance considered. Figure 5.2 compares their vehicle count results when the lexicographic objective and the central depot configuration are used on the clusters inside the city, whereas Figure 5.3 does the same for the clusters outside. The horizontal axis lists the problem instances while the vertical displays the vehicle counts. Figure 5.2 shows a typical trend: The DARP procedure produces stronger lower bounds and the CTSPAV procedure produces smaller vehicle counts. This trend carries over to Figure 5.3. Even though there are a few instances where the DARP procedure outperforms the CTSPAV procedure, overall, the CTSPAV matches or outperforms the DARP on more than 80% of the instances shown in the figure. The vehicle count results of the instances with the local depot configuration exhibit similar trends and are not shown here.

Table 5.1 then compares the aggregated vehicle count results from all the clusters for the same lexicographic objective and central depot configuration. This comparison is especially important as the results of all the clusters are summarized through aggregation in the case study. The table shows a significant difference between the results of the two procedures, with the DARP procedure producing aggregated vehicle counts that are 45% larger inside the city, 6% larger outside the city, and 11% larger overall. This significant difference makes another strong case for using the results of the CTSPAV procedure in the case study.

Table 5.1: Difference in Aggregate Vehicle Counts of CTSPAV and DARP Procedures

| Location | Aggregate vehicle count | | |
|---|---|---|---|
| | CTSPAV procedure | DARP procedure | Percentage difference |
| Inside | 78 | 113 | +45% |
| Outside | 652 | 694 | +6% |
| Combined | 730 | 807 | +11% |

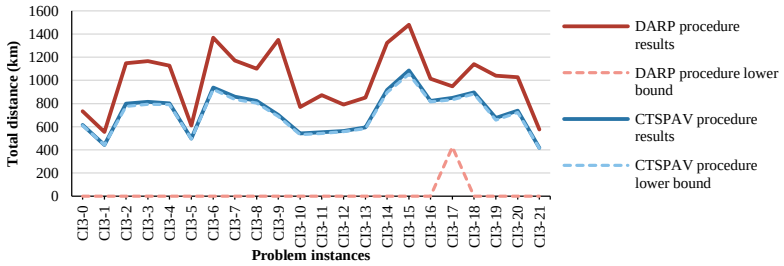



Figure 5.4: Comparison of Total Distance Results for Problem Instances Inside City Limits with the Distance-Minimization Objective and the Central Depot Configuration.

Figures 5.4 and 5.5 compare the results when minimizing total distance under the central depot configuration for the clusters inside and outside the city respectively. The horizontal axis lists the problem instances while the vertical displays the total distances. For a few clusters outside the city, even a single column-generation iteration of the DARP procedure cannot be completed within the time limit. The comparison for these instances are therefore excluded from Figure 5.5. Similar to Figures 5.2 and 5.3, the CTSPAV procedure produces stronger total distance results. In fact, it outperforms the DARP procedure in all the problem instances considered. In addition, the procedure also appears to produce stronger lower bounds for this objective function. There is a caveat to this observation however. The weak lower bounds of the DARP procedure in these cases can be attributed mainly to its column-generation phase not converging within its (longer) time limit. When the column generation does converge, e.g., in instances CO3-51 and CO3-55 in Figure 5.5, the lower bounds produced are comparable to those of the CTSPAV procedure. The figures also highlight the excellent optimality gap of the CTSPAV procedure for this objective function. Once again, the total distance results of instances with the local depot configuration are not summarized here as they display similar trends. Since the CTSPAV procedure consistently produces the stronger final results, it is used to obtain the results for the subsequent case study.

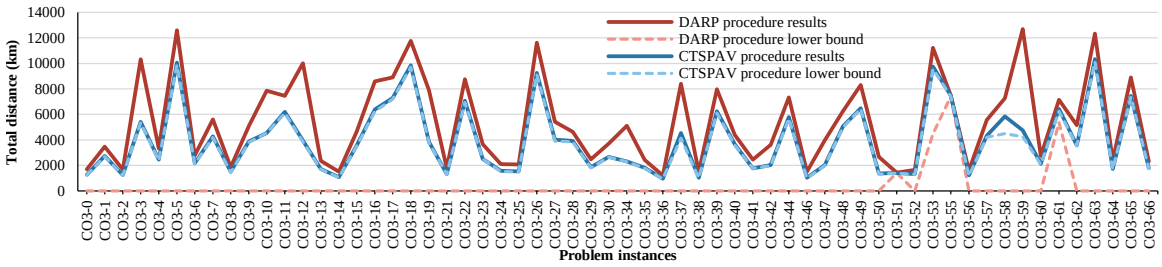



Figure 5.5: Comparison of Total Distance Results for Problem Instances Outside City Limits with the Distance-Minimization Objective and the Central Depot Configuration.

### 5.5.4 Vehicle and Travel Distance Reduction Results

Figures 5.6 and 5.7 show aggregated vehicle count (VC) results of all clusters inside and outside the city limits respectively for the first four weekdays of week 2. Each figure shows the VC results for every combination of objective function (lexicographic or distance minimization) and depot configuration (central or local) for the CTSPAV: they are labeled "Lex Central", "Dist Central", "Lex Local", and "Dist Local". Each figure also shows the VC results of the trips under the existing no-sharing conditions and of the CTSP described in Chapter 3 for additional perspective. The percentages of the VCs for each method as a fraction of the no-sharing VC are also included. The figures indicate that the CTSPAV consistently requires fewer vehicles than the CTSP to cover all the trips regardless of the location of the clusters, the objective function, or the depot configuration. This is not surprising, as the CTSPAV addresses the key limitation of the CTSP. As mentioned in Chapter 3, the routes of the CTSP are relatively short as the number of locations they can visit are limited by the ride-duration constraints of their drivers as well as by the time windows at the origins and destinations of the driver trips. By contrast, AVs are not subject to these limitations. Therefore, they can travel back and forth between the parking structures and the neighborhoods to serve trips throughout the day, consequently requiring fewer vehicles to cover the same number of trips. *What is striking, however, is the magnitude of the reduction in the number of vehicles: The VCs are reduced by 96% and 90% inside and outside the city limits respectively when using AVs.*

The results for the different objective functions show that the vehicle reduction of the lexicographic objective is significantly better than that of the distance minimization. Again, this result is not surprising as reducing VC is the primary goal of the former objective function, whereas it is not a consideration in the latter. The discrepancy between the vehicle reductions inside and outside the city limits can be attributed to the larger distance between the parking structures and the neighborhoods outside the city. As a result, the AVs serving these neighborhoods spend a larger fraction of their time in the transit phase, therefore

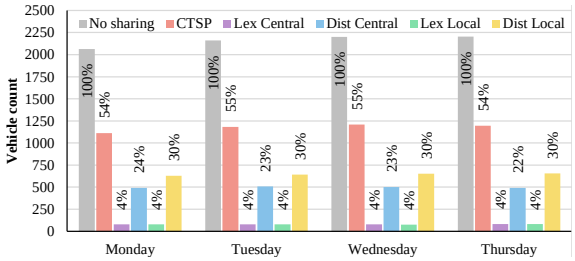



Figure 5.6: Aggregate Vehicle Count Results from All Clusters Inside City Limits.

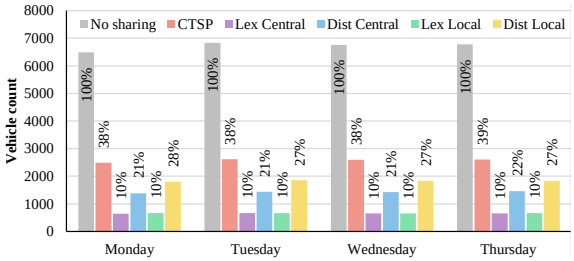



Figure 5.7: Aggregate Vehicle Count Results from All Clusters Outside City Limits.

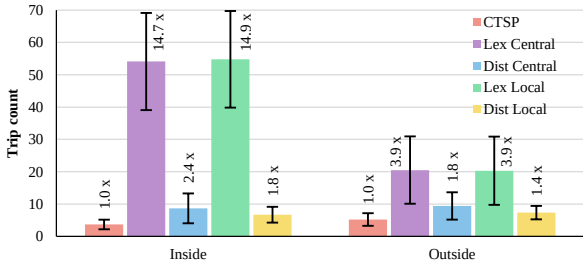



Figure 5.8: Average Number of Trips Served by Routes of Each Method.

limiting the number of trips they could serve in a day. This factor is further highlighted in Figure 5.8 which summarizes the average number of trips served by the routes of each method. The figure also shows each count as a multiplicative factor of the count for the CTSP, and the error bars depict the standard deviations of the trip counts. For the CTSPAV with the lexicographic objective, the routes from the clusters outside the city visit significantly fewer nodes than those inside the city as their transit phases are longer. However, regardless of the position of the clusters, the routes of the CTSPAV consistently cover more trips on average than those of the CTSP. *In fact, inside the city limits, the routes of the CTSPAV with the lexicographic objective serve, on average, an order of magnitude more trips than those of the CTSP.*

Figures 5.9 and 5.10 summarize the corresponding aggregated vehicle miles traveled (VMT) for all clusters inside and outside the city limits respectively. Similar to Figures 5.6 and 5.7, they show the CTSPAV results for every combination of objective function and depot configuration, as well as the results of the CTSP and of the trips under the no-sharing conditions for additional perspectives. Similarly, the percentages represent the VMT of each method as a fraction of the no-sharing VMT. The VMT percentage of each method outside the city limits is consistently smaller than those inside. This can be attributed to the neighborhoods outside the city being further away from the parking structures.

When a central depot is used for the CTSPAV, the VMT is reduced by 15–19% inside the city (resp., 30–34% outside the city). Although sizeable, the reduction is not as

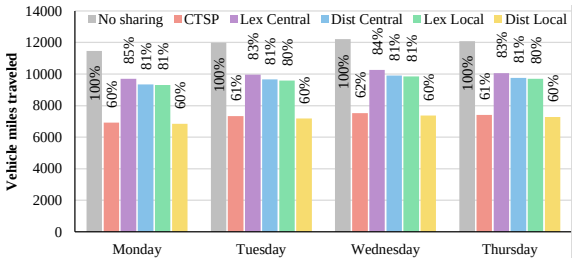



Figure 5.9: Aggregate Vehicle Miles Traveled from All Clusters Inside City Limits.

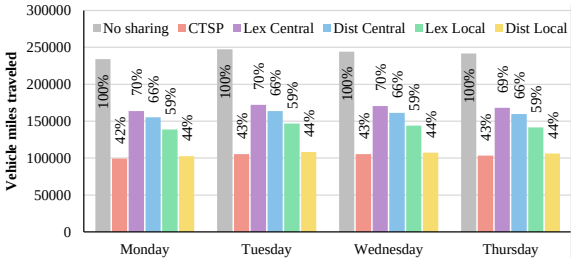



Figure 5.10: Aggregate Vehicle Miles Traveled from All Clusters Outside City Limits.

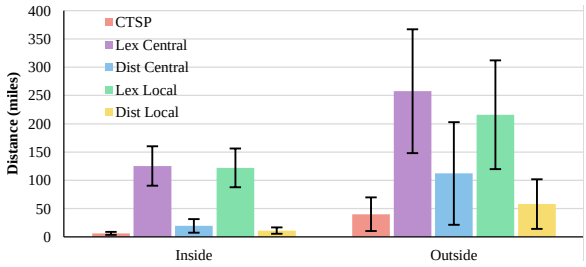



Figure 5.11: Average Distance Traveled Per Vehicle of Each Method.

significant as that of the CTSP, as the AVs have to travel back and forth between the neighborhoods and the parking structures which increases their total travel distance. When the different objective functions for the central depot configuration are compared, the distance-minimization objective only improves the VMT by 2–4% at the expense of significantly larger VCs. The VMT is further reduced when local depots are used, as each AV has to travel a shorter distance from the depot to reach its first pickup location and from its last drop-off location back to the depot since both these locations are in the neighborhoods. When the different objective functions for this depot configuration are compared, it can be seen that the distance-minimization objective significantly improves the VMT by 15–21%, but once again at the expense of a significant increase in VC. In fact, the VMT of the distance-minimization objective is comparable to that of the CTSP. This can be explained by referring to Figure 5.8 which shows their trip counts to be similar. In other words, the VMT of the CTSPAV with the distance-minimization objective and the local depot configuration is comparable to the CTSP because its routes are serving fewer trips per day.

Figure 5.11 provides further insight into the daily average of the distance traveled per vehicle for each method. The error bars depict the standard deviations for each method. As expected, the average travel distances are larger for the clusters outside the city for each method. They are also larger for the CTSPAV when the lexicographic objective is used as the AVs travel back and forth more between the neighborhoods and parking structures in order to reduce the vehicle count.

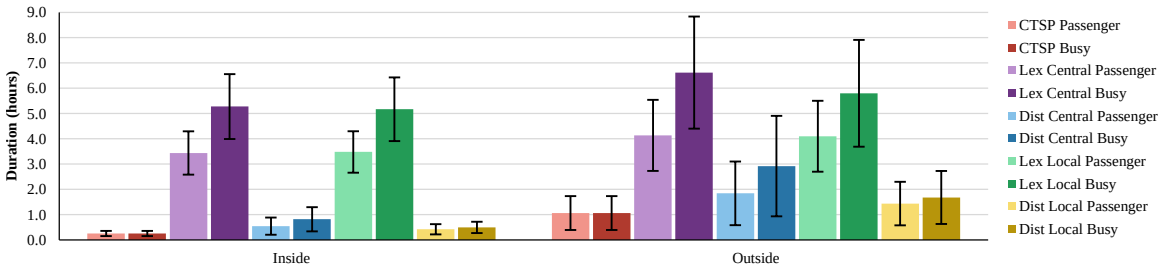



Figure 5.12: Average Passenger and Busy Ride Durations of the Routes of Each Method.

Figure 5.12 takes a look at the daily average of passenger and busy ride durations. The error bars again represent the standard deviations for each method. The passenger duration is defined as the total duration of the day during which at least one passenger is on the vehicle, whereas the busy duration is the passenger duration of a vehicle combined with the duration spent traveling between locations with no passengers. The complement of the busy duration, the idle duration, is therefore the duration of the day during which a vehicle is parked at a depot/at home/at a parking structure, combined with the duration spent waiting to pick up passengers while being empty. The passenger and busy durations of the CTSP are therefore identical as its vehicles are driven by the commuters themselves. In other words, they never travel without any passengers. For the CTSPAV, its busy duration is longer than its passenger duration as the latter is a subset of the former. In fact, the difference between the two represents the duration spent by the vehicle traveling without any passengers. For the lexicographic objective, this duration takes up a large fraction of its busy time, which supports earlier claims that the vehicles under this configuration do more back-and-forth traveling to reduce its vehicle count.

The average durations of each method for the clusters outside the city limits are longer than the corresponding durations inside as the vehicles must travel farther between the neighborhoods and the parking structures. The CTSP also has the shortest passenger and busy durations, which are consistent with the results from Figure 5.8 which showed the CTSP having the lowest average trip count. For the various configurations of the CTSPAV, the ones utilizing the lexicographic objective produce the largest passenger and busy durations, also consistent with their average trip-count results from Figure 5.8. The results also reveal another drawback of the distance-minimization objective. Not only does the configuration require more vehicles to cover the same number of trips, but its vehicles are less busy throughout the day than those of the lexicographic objective. In other words, the vehicles spend a longer time every day being idle, which somewhat defeats the purpose of utilizing AVs in the first place.

Figures 5.13 and 5.14 provide a deeper look into how the passenger durations are spent

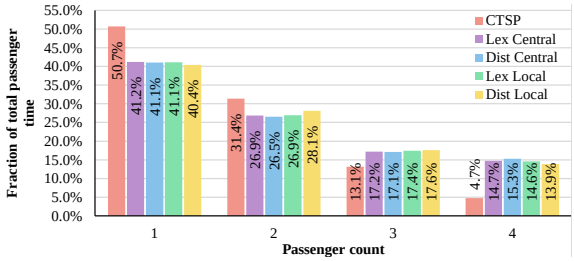



Figure 5.13: Fraction of Total Passenger Time Spent Serving 1, 2, 3, and 4 Passengers for Clusters Inside City Limits.

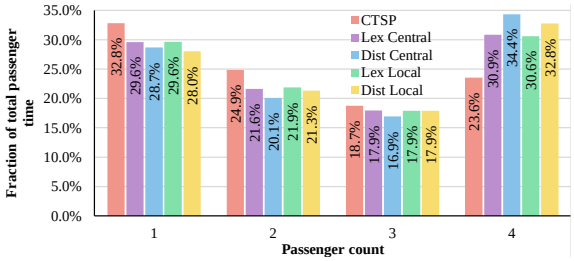



Figure 5.14: Fraction of Total Passenger Time Spent Serving 1, 2, 3, and 4 Passengers for Clusters Outside City Limits.

by the routes of each method for clusters inside and outside the city limits respectively. It shows the fraction of the total passenger duration spent serving one, two, three, or four passengers for each configuration. Inside the city, these fractions progressively decrease as the number of passengers increases. However, the CTSPAV spends a larger fraction of its time serving three or four passengers than the CTSP, again reinforcing the advantage of the CTSPAV. This is due to two factors. First, the mini routes in the CTSP must start and finish with the same driver. Second, the CTSP must also synchronize the inbound and outbound routes, since the same drivers are used for both. Because the schedules of the passengers very often differ, some of the mini routes cannot be used by the CTSP but can by the CTSPAV. This observation is carried over to the clusters outside the city, whereby every configuration of the CTSPAV spends most of its time serving four passengers, while the same cannot be said for the CTSP. This observation can be attributed to the longer transit phase of the routes for the clusters outside the city combined with the vehicles being used to their full capacity during the transit phase by the CTSPAV.

In summary, the lexicographic objective for the CTSPAV has the greatest vehicle reduction potential, and the configuration of the depot does not appear to affect this potential. Its routes consistently cover the largest number of trips on average, and they also have the longest passenger and busy durations, which are all desirable characteristics for an effective AV trip-sharing platform. The local depot configuration for the lexicographic objective does produce slightly better VMT results. This small benefit, however, may be outweighed by the logistical benefits of having a central depot, e.g., the convenience and the cost effectiveness of having a central location for maintaining and refueling/recharging all AVs. The CTSPAV with the distance-minimization objective and the local depot configuration consistently produces the lowest VMT, however, as mentioned earlier, the result is obtained at the expense of higher vehicle counts. Besides that, as shown in Figures 5.8 and 5.12, the AVs also serve relatively fewer trips and spend more of their time being idle every day

under this configuration.

### 5.5.5 Cost Analysis

This section reports a simple, coarse analysis to estimate the cost of operating the trip-sharing platform over a 5- and 10-year period. The analysis is not intended to be a sophisticated or complete measure of the total cost of the platform per se; instead, it is aimed to obtain a rough understanding of the trade-off between vehicle and operating costs of the platform. The analysis focuses only on the CTSPAV with the central depot configuration, as it is preferred over the local depot configuration due to its aforementioned logistical benefits. The analysis first considers vehicle-related costs (referred to simply as vehicle costs) and then operating costs.

The vehicle cost is age related and consists of the vehicle depreciation over $y$ years and a distance-related cost. This last cost consists of an average fuel cost of \$0.15 and an average vehicle-value depreciation of \$0.08 per mile traveled. An exponential decay function is used to model how a vehicle value depreciates with age. More precisely, a vehicle's depreciation $\gamma$ over $y$ years is given by:

$$\gamma = p - p(1 - \nu)^y \tag{5.31}$$

where $p$ is the vehicle's initial price and $\nu$ is its annual depreciation rate. This analysis uses a depreciation rate of 24% for the first year and 15% for subsequent years. The total vehicle cost over $y$ years is then obtained by multiplying the depreciation $\gamma$ of every vehicle over that period with the VC required to cover all daily trips, and then multiplying the distance cost with the average daily VMT over the time period considered, assuming trips are only made on weekdays.

Figures 5.15 and 5.16 show the results of the vehicle-cost analysis over 5 and 10 years for clusters inside and outside the city limits respectively. In each figure, the horizontal axis displays the range of possible initial prices of an AV, whereas the vertical axis displays the corresponding total vehicle cost. The figures show that, for the range of initial prices considered, the total vehicle cost is always dominated by the contributions from the vehicle-age cost. Therefore, the distance-minimization objective that requires larger VCs is always more expensive than the lexicographic objective, and this is true regardless of the location of the clusters or the time period considered for the analysis.

The operating cost considers the average annual cost of a parking permit for each vehicle combined with an estimated fixed cost for installing a charging station for each AV. For this analysis, an annual parking permit cost of \$800 is used together with a charging

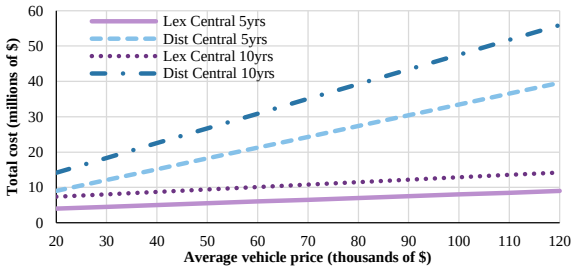



Figure 5.15: Total Vehicle Cost for CTSPAV Platform Inside City Limits Over 5 and 10 Years.

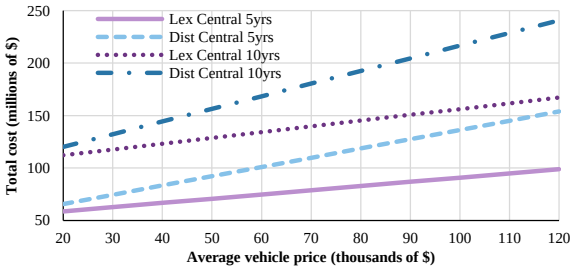



Figure 5.16: Total Vehicle Cost for CTSPAV Platform Outside City Limits Over 5 and 10 Years.

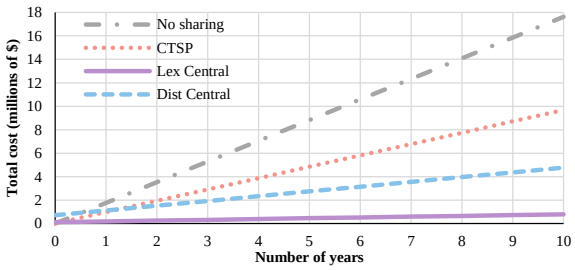



Figure 5.17: Total Operating Cost for CTSPAV Platform Inside City Limits Over 10 Years.

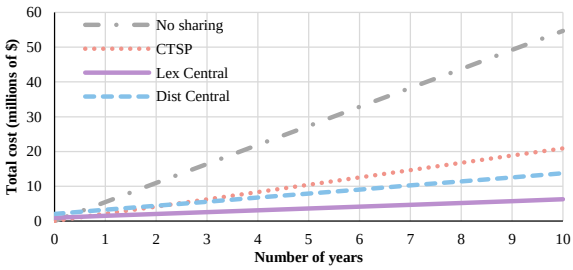



Figure 5.18: Total Operating Cost for CTSPAV Platform Outside City Limits Over 10 Years.

station installation cost of $1,400 per vehicle. The total operating cost over $y$ years is then simply calculated by multiplying the charging station installation cost with the VC required to cover all daily trips, and then multiplying the annual parking permit cost with the VC and the number of years.

Figures 5.17 and 5.18 display the results of the total operating cost as a function of the number of years for the clusters inside and outside the city limits respectively. The total parking cost of the CTSP and of the vehicles under the no-sharing condition are also included for additional perspective. Since there are no fixed costs associated with the CTSP or with the no-sharing condition, their operating costs are lower than those of the CTSPAV in the beginning. However, as time increases, the parking costs start to dominate, causing the methods with larger VCs to be more expensive. In fact, inside the city limits, both the CTSPAV methods considered become cheaper as early as the second year, while the same happens as early as the third year outside the city. The gap in operating costs between the different methods considered also grows with time and highlights the cost effectiveness of the CTSPAV with the lexicographic objective as it uses the fewest number of vehicles.

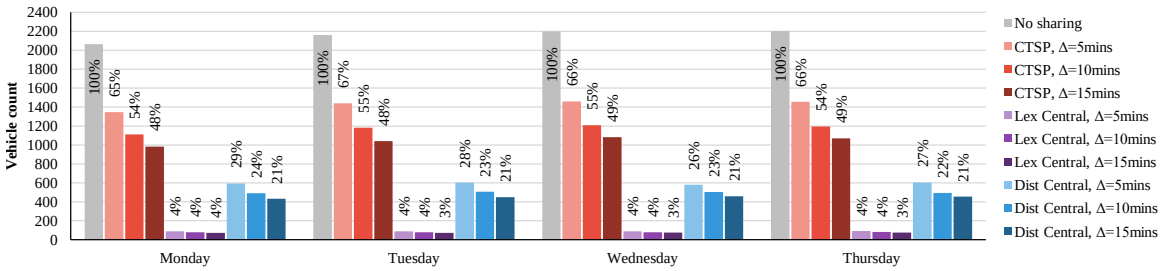



Figure 5.19: Aggregate Vehicle Count Results Inside City Limits for $\Delta \in \{5, 10, 15\}$ mins.

### 5.5.6 Sensitivity to $\Delta$

The parameter $\Delta$ represents the maximum amount of time by which each rider needs to shift (up or down) her desired arrival and departure times at a parking structure. It has a direct impact on the QoS of the riders, and it is desirable to have $\Delta$ be as small as possible. Limiting its value however restricts the flexibility of the schedules and could negatively impact trip shareability. To study the impact of varying $\Delta$ on the results of the CTSPAV, the procedure with the central depot configuration is applied on every cluster with $\Delta$ set to $\{5, 15\}$ mins. The results are then compared against those of $\Delta = 10$ mins. Figures 5.19 and 5.20 compare aggregated VCs of all clusters inside and outside the city limits respectively. Figures 5.21 and 5.22 then compare aggregated VMT from all the clusters inside and outside the city respectively. Sensitivity results of the CTSP are also included in the figures, as well as the results of the no-sharing condition and the percentage of each quantity as a fraction of the no-sharing results for additional perspective.

The results show that reducing $\Delta$ to 5 mins adversely affects the vehicle reduction capability of the CTSP, reducing its VCs by approximately 12% inside the city (resp. 8% outside the city). Increasing $\Delta$ to 15 mins improves vehicle reduction by approximately 6% inside the city (resp. 4% outside the city). This is the evidence of a trade-off between QoS and trip shareability. By contrast, the VC results of the CTSPAV with the lexicographic objective exhibit very little sensitivity to $\Delta$, whereby the VCs change by $\leq 1\%$ as $\Delta$ is varied by $\pm 5$ mins. *This bodes very well for the CTSPAV as it indicates that a reduction in $\Delta$ to improve the QoS for the riders will not have a significant impact on its vehicle reduction capability*. Finally, the VCs of the CTSPAV with the distance-minimization objective display a modest sensitivity to $\Delta$, whereby decreasing $\Delta$ by 5 mins degrades the VCs by approximately 5% inside the city (resp. 3% outside the city), and increasing it by 5 mins improves the VCs by approximately 2% insde the city (resp. 1% outside the city).

The sensitivity analysis on the aggregated VMT paints a different picture, whereby every method considered exhibits comparable sensitivity to the variations in $\Delta$. The results

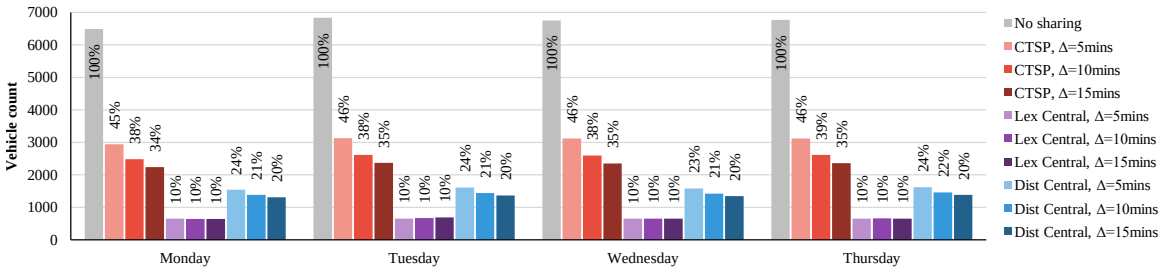



Figure 5.20: Aggregate Vehicle Count Results Outside City Limits for $\Delta \in \{5, 10, 15\}$ mins.

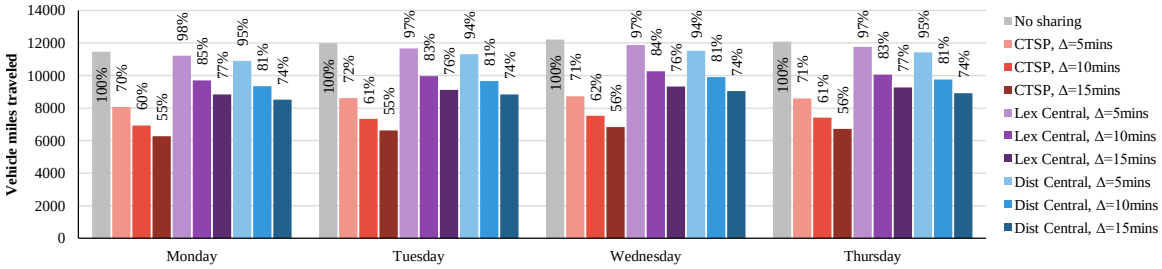



Figure 5.21: Aggregate Vehicle Miles Traveled Inside City Limits for $\Delta \in \{5, 10, 15\}$ mins.

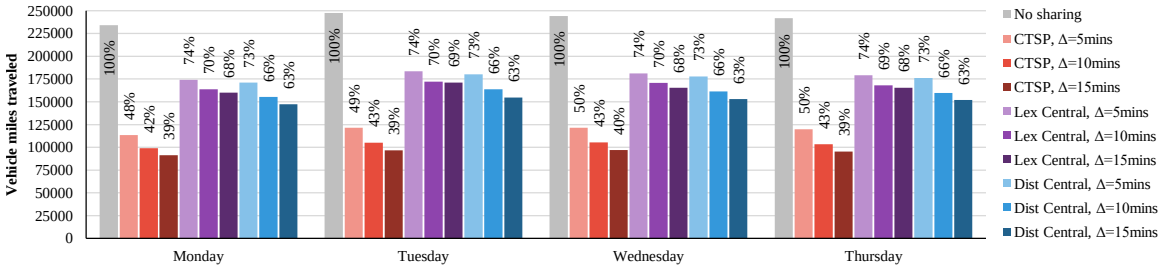



Figure 5.22: Aggregate Vehicle Miles Traveled Outside City Limits for $\Delta \in \{5, 10, 15\}$ mins.

once again display a trade-off, this time between QoS and travel-distance reduction, which is evident from the VMT decreasing as $\Delta$ is increased and vice versa. The increase in VMT of the CTSPAV, regardless of the objective function or the position of the clusters, when $\Delta$ is reduced stems from the reduction of opportunities for trip aggregation as a result of the tighter time windows. The AVs would therefore need to increase their back-and-forth traveling between the parking structures and the neighborhoods to serve the same number of trips, leading to the increases in their travel distance.

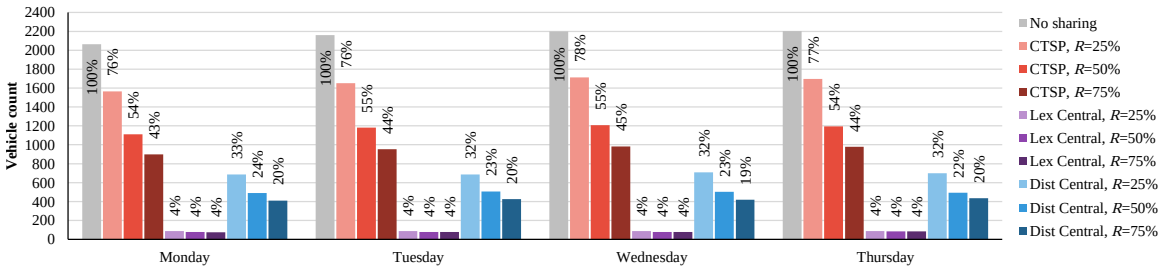



Figure 5.23: Aggregate Vehicle Count Results Inside City Limits for $R \in \{25\%, 50\%, 75\%\}$.

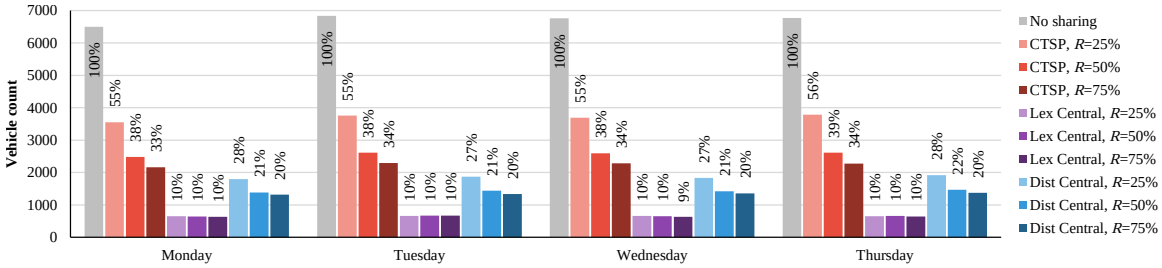



Figure 5.24: Aggregate Vehicle Count Results Outside City Limits for $R \in \{25\%, 50\%, 75\%\}$.

### 5.5.7 Sensitivity to $R$

The parameter $R$ is yet another parameter that may influence the QoS of the riders, as it directly influences the maximum amount of time every rider spends in a vehicle. Limiting the value of $R$ improves the QoS as it leads to shorter ride durations. However, it also results in less flexible trip schedules, which could consequently reduce the potential for trip aggregation. Therefore, one would anticipate a trade-off between QoS and trip shareability when varying $R$ that is similar to that observed when varying $\Delta$. To investigate this trade-off, a sensitivity analysis is conducted by setting $R \in \{25\%, 75\%\}$, applying the CTSPAV procedure with the central depot configuration on every cluster, and comparing the results with those of $R = 50\%$.

Results of the analysis on VCs inside and outside the city limits are summarized in Figures 5.23 and 5.24 respectively, whereas the analysis on VMT inside and outside the city limits are displayed in Figures 5.25 and 5.26 respectively. Similar to previous analyses, results from the original CTSP, from the no-sharing condition, and the percentages of each quantity as a fraction of the no-sharing results are included for reference.

The VC results of the CTSP display the expected trade-off described earlier where they appear to be very sensitive to variations in $R$. Vehicle reduction significantly degrades by approximately 22% inside the city (resp. 17% outside the city) when $R$ is reduced to

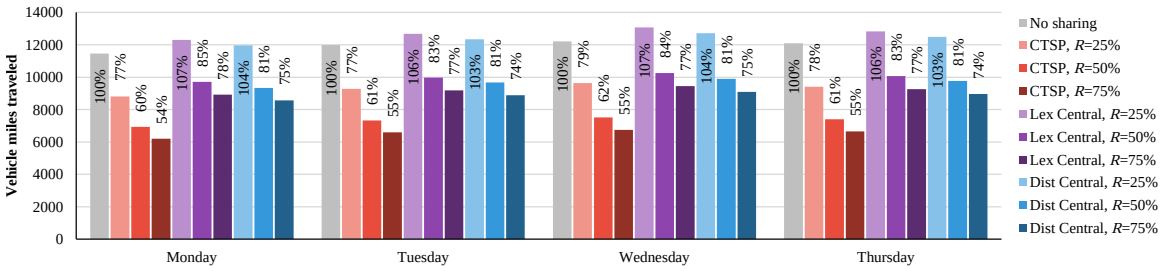



Figure 5.25: Aggregate Vehicle Miles Traveled Inside City Limits for $R \in \{25\%, 50\%, 75\%\}$.

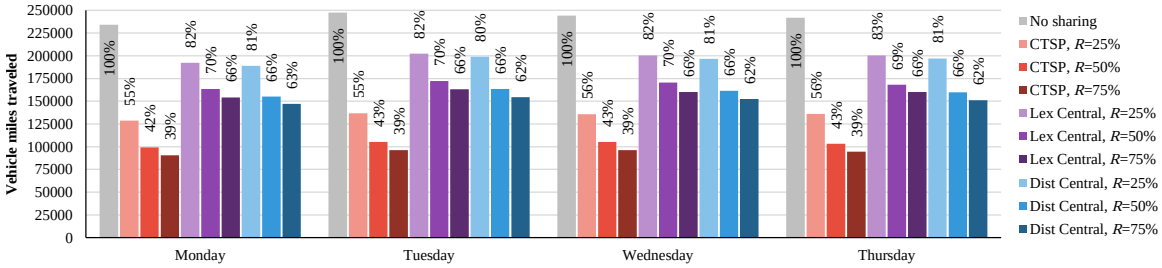



Figure 5.26: Aggregate Vehicle Miles Traveled Outside City Limits for $R \in \{25\%, 50\%, 75\%\}$.

25%, and it improves by approximately 11% inside the city (resp. 5% outside the city) when $R$ is increased to 75%. This indicates that attempts to reduce the maximum ride duration will result in significantly reduced trip shareability in the CTSP. By contrast, such a drawback is not evident from the results of the CTSPAV with the lexicographic objective, as its VCs exhibit very little sensitivity to the changes in $R$. The results show changes that are $< 1\%$ both inside and outside the city as $R$ is varied by $\pm 25\%$[2]. *This result provides another positive outlook for the CTSPAV as it promises that the vehicle reductions will not be adversely affected by attempts to increase the QoS by reducing the maximum ride durations of the riders.* Finally, the VC results of the CTSPAV with the distance-minimization objective display a relatively modest sensitivity to $R$, whereby decreasing $R$ to 25% increased the VCs by approximately 9% inside the city (resp. 6% outside the city), and increasing $R$ to 75% reduced the VCs by approximately 3% inside the city (resp. 1% outside the city).

Analysis of the VMT results reveals different observations. Inside the city limits, both objective functions of the CTSPAV appear to be more sensitive to changes in $R$ than the CTSP; decreasing $R$ to 25% leads to approximately a 23% increase in VMT for both ver-

[2]For the clusters outside the city limits, the column-generation phase generated on average 34% more columns for $R = 75\%$ than it did for $R = 50\%$, which caused the MIP to be significantly harder to solve. The MIP time limit for these instances is therefore extended to 2 hrs to account for this increase in complexity.

sions of the CTSPAV (compared to a 17% increase for the CTSP). In fact, the increase is so significant in this case that their aggregated values exceed those for the no-sharing condition. This indicates that the opportunities for trip aggregation in this case is diminished to the point that any savings in travel distance is overshadowed by a greater increase in back-and-forth traveling resulting from having to cover the trips with approximately the same number of vehicles. Outside the city limits, the VMT of every method exhibit comparable sensitivity to changes in $R$, once again displaying the trade-off between QoS and travel-distance reduction.

### 5.5.8 Effect of Increasing Vehicle Capacity

This section explores the effects of increasing vehicle capacity, $K$, on the results of the CTSPAV to investigate if there are any benefits to be gained from using larger vehicles. The results are obtained by varying $K$ between 1 to 8 and applying the CTSPAV procedure with the central depot configuration on all trips made on the Wednesday of week 2.[3]Figure 5.27 shows aggregated VCs from all clusters for every $K$ value, while Figure 5.28 does the same for aggregated VMT. The percentages in both figures indicate each quantity as a fraction of the no-sharing values.

Both figures reveal that marginal improvements (reductions) to the VCs and the VMT decrease with increasing $K$. In fact, they show that almost no improvement is obtained for both the VCs and the VMT beyond $K = 5$, and the biggest improvement is obtained when increasing $K$ from 1 to 2. These results are consistent with the findings from Farhan and Chen (2018) which focused on an SAV system for on-demand trips, whereby they discovered that the decreases to the fleet size was marginal for $K > 2$. Alonso-Mora et al. (2017), who studied the potential of a ride-sharing optimization algorithm that is also tailored for on-demand trip requests, also discovered diminishing marginal improvements to mean travel delay, mean waiting time, and mean travel distance when $K$ is increased from 4 to 10. However, they did observe marginally high vehicle occupancy during peak hours for $K = 10$, whereby approximately 10% of the vehicles had eight or more passengers. The discrepancy between their results and ours is easily attributable to the enormous disparity in the spatio-temporal density of the trips considered in both works, where the NYC

[3]To accommodate the exponential increase in problem complexity that results from the increase in $K$, the following changes were applied to the CTSPAV procedure for $K > 4$. A 10-minute timeout is applied to the label-setting dynamic program of the $\text{PSP}_{\text{CTSPAV}}$. The algorithm is terminated if it exceeds the timeout and it just returns all complete, non-dominated mini routes discovered at that point with negative reduced costs. The column-generation procedure is also seeded with all feasible mini routes covering up to 2 trips which are obtained from an exhaustive search procedure. Finally, the time budget for the column-generation phase is extended to 2 hours.

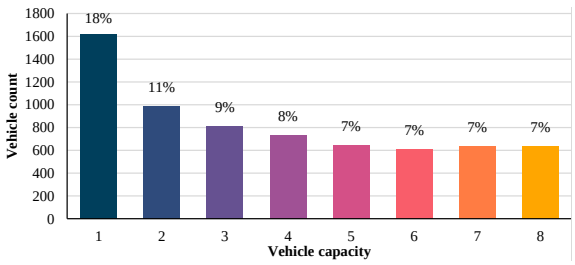



Figure 5.27: Effect of Increasing Vehicle Capacity on Aggregated Vehicle Count.

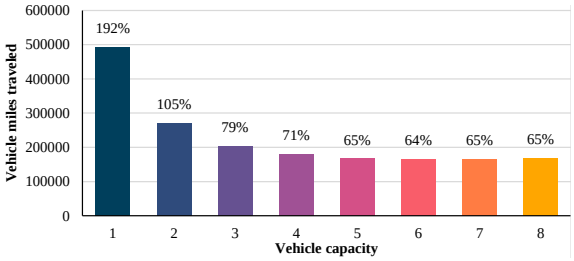



Figure 5.28: Effect of Increasing Vehicle Capacity on Aggregated Vehicle Miles Traveled.

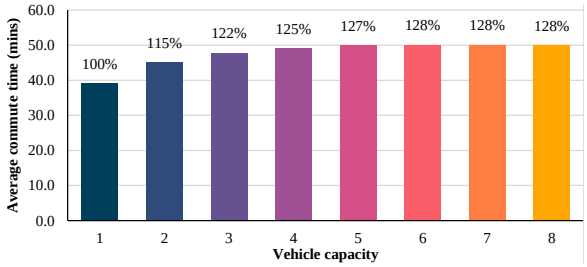



Figure 5.29: Effect of Increasing Vehicle Capacity on Average Commute Time.

taxicab dataset (NYC Taxi & Limousine Commission 2020) used in their work had a daily average of 450,000 trip requests (25 times more than the 18,000 inbound and outbound trips from our dataset) taking place on the island of Manhattan, NYC, which spans only 23 square miles (an area that is more than 550 times smaller than the 13,000-square-mile region considered in our dataset). The immense spatio-temporal density of the trips from the NYC taxicab dataset provides significantly more trip aggregation opportunities by increasing the availability of trips with compatible itineraries throughout the day, a factor that is critical for maximizing the utilization of larger vehicle capacities. It must also be noted that the marginal benefits obtained from increasing $K$ in this work is accompanied by an exponential increase in problem complexity (in fact, the very slight increase in the VCs and the VMT for $K > 6$ can be attributed to the increase in the problem complexity and the procedure not being able to find better solutions even with the extended time budget).

Finally, Figure 5.29 shows the effect of increasing $K$ on the average ride duration per commuter. The percentages represent each quantity as a fraction of the value when $K = 1$. Unsurprisingly, the best average commute time is obtained when $K = 1$ as the rides are not shared under this setting, and the times increase with increasing $K$ due to the corresponding increase in ride sharing. The diminishing nature of the marginal increases in average commute time as $K$ is increased also appears to mirror the marginal decreases in the VCs or the VMT. The figure also shows that when $K = 4$, the average commute time

only increases by 25% (even though $R = 50\%$), which bodes very well for maintaining the QoS of the riders of the CTSPAV.

## 5.6 Conclusion

This work originated from the desire to understand the potential benefits of autonomous vehicles on a car-pooling platform that maximizes ride sharing for the commute trips of a community. The central problem powering the platform is the CTSP: It was originally introduced in Chapter 2 to reduce parking utilization and traffic congestion in urban areas by leveraging the structure of commuting patterns and urban communities. The CTSP was shown to reduce the number of vehicles significantly in a real case study.

Given that the vehicles are idle for most of the day by the definition of car pooling, it is interesting to study how much additional reduction in fleet size would come from using autonomous vehicles, as well as its impact on the miles traveled. To answer this question, this chapter defined the CTSPAV and proposed two column-generation procedures to obtain high-quality solutions. The first approach (the CTSPAV procedure) assembles feasible mini routes into an overall routing plan, while the second approach (the DARP procedure) reduces the CTSPAV into a DARP. The optimization problems considered (1) a lexicographic objective that first minimizes the required vehicle count and then their total distance, and (2) a distance-minimization objective that just minimizes the total travel distance.

The CTSPAV was evaluated on the large-scale, real-world dataset of commute trips from the city of Ann Arbor, Michigan, which contains detailed information for an average of 9,000 daily commute trips over a month. The experimental results revealed that the CTSPAV procedure with the lexicographic objective reduces daily vehicle usage by 92% while at the same time also reducing vehicle miles traveled by 30%. Its vehicle reduction results represent a 34% improvement relative to that of the CTSP. Examining the solutions showed that the CTSPAV generates significantly longer routes by traveling back and forth between the communities and the commuting destination. A cost analysis also showed that fleets of autonomous vehicles are eminently viable from an economic standpoint in this setting. Finally, sensitivity analyses revealed that the vehicle count results of the lexicographic objective are more resilient to variations in the size of the time windows and the length of the ride-duration limits of the trips. The number of vehicles required to cover all the trips changed by less than 1% as the two parameters are varied, while the same cannot be said about the CTSP.

# CHAPTER 6

# Commuting with Autonomous Vehicles: A Branch-and-Cut Algorithm with Redundant Modeling

## 6.1 Introduction

This work is the culmination of a four-year study on the benefits of ride-sharing and car-pooling platforms for serving commuting needs. It was originally motivated by the desire to relieve parking pressure in the city of Ann Arbor, Michigan. Parking structures are expensive and are often located in prime locations for the convenience of commuters. In Ann Arbor, the parking pressure was primarily caused by commuters to the University of Michigan, the city's largest employer which has more than 50,000 employees.

Detailed information about the commuting patterns of these employees was gathered by recording trip data from approximately 15,000 drivers who use the 15 university-operated parking structures located in the downtown area over the month of April 2017. The data consisted of the exact arrival and departure times of every commuter to and from the parking structures, which were then joined with the precise locations of the parking structures and the home addresses of every commuter to reconstruct their daily trips. The dataset revealed several intriguing temporal and spatial characteristics. First, the peak arrival and departure times, which are depicted in Figure 1.2 for the weekdays of the busiest week, coincide with the typical peak commuting hours. Second, the strong consistency of the trip schedules was seen as a significant opportunity for car-pooling and ride-sharing platforms. Third, the commuting destinations (the parking structures) are located within close vicinity of each other in the downtown area (as they are university-owned structures), whereas the commuting origins (the commuter homes) are located in the neighborhoods surrounding the downtown area, as well as in Ann Arbor's neighboring towns. This spatial structure, which is quite typical in many American cities, was also seen as an opportunity for trip-sharing

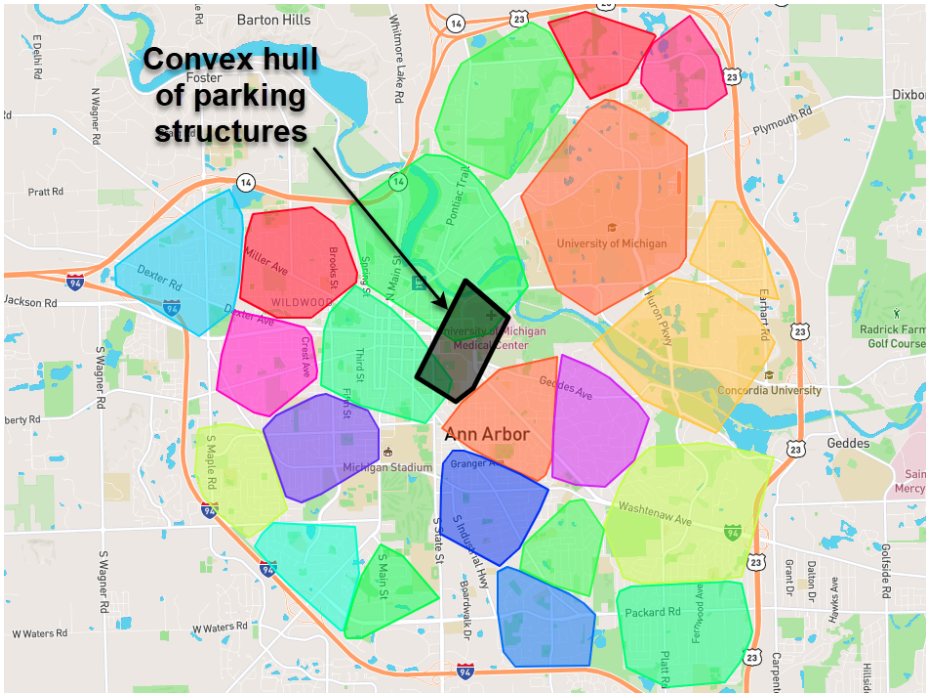



Figure 6.1: Convex Hulls of Artificial Neighborhoods Resulting from Clustering Algorithm

platforms.

With this in mind, Chapter 3 formalized the key optimization problem faced by a car-pooling platform that would serve commute trips and called it the CTSP. More precisely, the CTSP conceptualizes the platform as a reservation-based system that receives the commute-trip requests—each consisting of a request to the workplace (inbound trip) and another to return back home (outbound trip)—ahead of time (e.g., the day ahead or the morning of each day). Each trip request includes small time windows for its departure and arrival, and each rider is guaranteed not to spend more than $R\%$ of her direct trip in commuting time. The CTSP was tailored for scenarios where: (1) The commuters travel to a common/centralized location, e.g., the commute trips of the employees of a corporate or university campus, or (2) The commuters live in a common/centralized location, e.g., the commute trips originating from a residential neighborhood or an apartment complex. These scenarios were inspired by the spatio-temporal structure observed in the Ann Arbor commute-trip dataset described earlier.

To implement such a platform and address the complexity of dealing with the massive volume of the trips from the dataset, we applied a two-stage approach:

1. It first clusters commuters into artificial neighborhoods based on the spatial proximity of their home locations using an unsupervised machine-learning algorithm;
2. It then finds the optimal routes for the commuters within each cluster.

Figure 6.1 provides an overview of the resulting clusters within Ann Arbor's city limits: It displays the convex hulls of the neighborhoods, as well as the convex hull of the centrally located parking structures. The optimization problem in step 2 is the CTSP: Each day, its goal is to use private vehicles owned by the commuters, select the set of drivers for the inbound and outbound routes of the vehicles, and design the routes in order to minimize the number of vehicles utilized, and hence the parking pressure. Solutions to the CTSP were shown to reduce daily vehicle usage for the Ann Arbor dataset by up to 57%.

Despite this significant potential, the results also highlighted several factors limiting further reductions in the daily vehicle counts. They included: (1) the nature of the CTSP routes that are typically short and (2) the necessity to synchronize the inbound and outbound routes since they must be performed by the same set of drivers. Indeed, as the drivers in the CTSP are selected from the commuters themselves, each route must begin and end at the origin and the destination of its driver. This bookending requirement subjects the total duration of the route to the temporal constraints of the driver's trip, restricting its length and consequently its ability to serve more trips. This, combined with the necessity of selecting an identical set of drivers for the inbound and outbound routes, limits the flexibility of the routes that can be generated and used in the CTSP routing plan.

The CTSPAV considered in this chapter was originally proposed in Chapter 5: Its goal was to overcome these shortcomings by leveraging AV technology that is lurking in the horizon. By removing the driver-related constraints, the CTSPAV was anticipated to allow the AV routes to be significantly longer than the CTSP routes. While these longer routes would significantly increase the number of commute trips that can be covered by each AV on any day, the algorithmic complexity for finding them was also expected to increase significantly. We therefore proposed a column-generation solution procedure, dubbed the CTSPAV procedure, that is a departure from the classical column-generation approach for solving typical VRPs. The latter typically entails solving a set-partitioning/covering master problem that ensures each customer is served, and a pricing subproblem that searches for feasible routes that depart from and return to a depot and have negative reduced costs. The CTSPAV procedure circumvents the anticipated complexity of searching for the long AV routes in the pricing subproblem by shifting some of the burden to the master problem and *exploiting the spatio-temporal structure of the dataset*. It leverages the spatial structure of the trips, whereby their origins and destinations are located in separate, disjoint "islands", which naturally decomposes the long AV routes into a sequence of shorter constituent routes by using a pricing subproblem that only searches for feasible "mini" routes with negative reduced costs. The mini routes are short by construction: Each covers only inbound or outbound trips exclusively, and each has distinct pickup, transit, and drop-off

phases during which it first visits only trip origins, then travels from an origin to a destination, and finally visits the trip destinations. These three phases are naturally encountered by each vehicle as it travels from a residential neighborhood to the workplace in the morning to serve inbound trips, and vice versa in the evening to serve outbound trips.

In order for the mini routes to be feasible, they must visit each location within a specified time window, ensure that the time spent on the vehicle by each rider does not exceed a specified limit, and cannot exceed the vehicle capacity. In other words, they must satisfy time-window, ride-duration, and vehicle-capacity constraints. Furthermore, they must also satisfy pairing and precedence constraints, which require a route visiting the origin of a trip to also visit its destination in the correct order. The master problem of the CTSPAV procedure is then responsible for stitching or chaining together the feasible mini routes to form longer AV routes that begin and end at a depot. In addition to ensuring that each trip is covered, the master problem must also select mini routes that are temporally compatible with each other, i.e., it needs to ensure that it is possible to travel from the last destination of one mini route to the first origin of another without violating the temporal constraints of the selected mini routes. All of this is done in service of a lexicographic objective function that first minimizes the number of formed AV routes (i.e., the vehicle count if each route is assigned to an AV) and then minimizes their total travel distance.

Since the routes of the CTSPAV satisfy time-window, ride-duration, capacity, pairing, and precedence constraints which are identical to those for the DARP (Cordeau and Laporte 2003a, 2007), the CTSPAV can be seen as a special version of the DARP that serves inbound-outbound trip pairs using AVs. In fact, any DARP algorithm can be used to solve the CTSPAV. Chapter 5 explored this possibility as well by investigating a DARP procedure for solving the CTSPAV. The procedure borrows heavily from an algorithm for the DARP proposed by Gschwind and Irnich (2015): It relies on the classical column-generation approach and uses a novel, label-setting dynamic program to solve its pricing subproblem. We discovered that, while the complexity of discovering the long AV routes in its pricing subproblem severely hampered the procedure's ability to find strong integer solutions within a time-constrained setting, the DARP procedure also produced superior primal lower bounds for the primary objective. On the other hand, the CTSPAV procedure produced stronger integer solutions within a similar time-constrained setting, but it does so at the expense of generating weaker lower bounds.

This work aims at addressing these limitations with two goals in mind:

1. To propose an exact algorithm for the CTSPAV;
2. To provide a conclusive and comprehensive analysis of the potential of the CTSPAV in reducing vehicle counts, travel distances, and traffic congestion.

To meet the first goal, we present an exact algorithm that improves upon the CTSPAV procedure from Chapter 5 by combining the insights from itself and the DARP procedure in a redundant modeling framework (Liberti 2004, Ruiz and Grossmann 2011). The proposed algorithm leverages the best characteristics of the CTSPAV and DARP procedures, i.e., the former's capability of producing strong integer solutions and the latter's ability of generating strong primal lower bounds. The *methodological contribution* of this work is to propose a branch-and-cut algorithm for solving the CTSPAV which leverages a novel dual-modeling technique. The branch-and-cut algorithm solves a mathematical model that exploits the spatio-temporal structure of the data, making it conducive to finding high-quality solutions quickly. At the same time, it also uses another mathematical model for the same problem to generate valid inequalities that are separated by a column-generation procedure to produce strong lower bounds. This work demonstrates the benefits of this dual-modeling approach through a comparison with a dedicated branch-and-cut procedure based on well-established families of valid inequalities, and with the heuristic column-generation procedure from Chapter 5.

The proposed exact branch-and-cut procedure is also embedded into an end-to-end approach which combines clustering and optimization to solve large-scale, real-world instances of the CTSPAV based on the Ann Arbor commute-trip dataset. This case study complements the methodological contribution by providing unique insights into the potential benefits of ride sharing and AVs for serving the commuting needs of many cities around the world. It demonstrates that a ride-sharing platform based on AVs can provide substantial reductions to daily vehicle counts and traffic congestion, as well as improve the traveled miles. In addition, this work contrasts, for the first time, the potential benefits and drawbacks of car-pooling and ride-sharing platforms along the same dimensions.

The rest of this chapter is organized as follows. Section 6.2 revisits the CTSPAV model from Chapter 5 and describes an algorithm for enumerating mini routes. Section 6.3 provides an overview of the branch-and-cut algorithm and covers the different families of valid inequalities considered in this work together with the heuristics used to separate them. Section 6.4 outlines how the algorithm is evaluated and presents the computational results. Section 6.5 documents the insights obtained from the case study. Finally, Section 6.6 provides some concluding remarks.

## 6.2 The Commute Trip-Sharing Problem for Autonomous Vehicles

This section revisits the specification of the CTSPAV which was first outlined in Chapter 5. The problem seeks a set of minimal cost AV routes to serve every inbound and outbound trip of a set of commuters, $\mathcal{C}$.

### 6.2.1 Notation

Let $n$ denote the total number of commuters, i.e., $n = |\,\mathcal{C}\,|$. For every commuter $i \in \mathcal{C}$, let nodes $i$, $n+i$, $2n+i$, and $3n+i$ represent the inbound pickup, inbound drop-off, outbound pickup, and outbound drop-off locations of the rider's trips respectively. Then let the sets of all inbound pickup, all inbound drop-off, all outbound pickup, and all outbound drop-off nodes be denoted by $\mathcal{P}^+ = \{1, \ldots, n\}$, $\mathcal{D}^+ = \{n+1, \ldots, 2n\}$, $\mathcal{P}^- = \{2n+1, \ldots, 3n\}$, and $\mathcal{D}^- = \{3n+1, \ldots, 4n\}$ respectively. Furthermore, let $\mathcal{P} = \mathcal{P}^+ \cup \mathcal{P}^-$ and $\mathcal{D} = \mathcal{D}^+ \cup \mathcal{D}^-$. With this notation, note that $n+i$ provides the drop-off node corresponding to any pickup node $i \in \mathcal{P}$. By definition of AV routes, the following precedence constraints apply to the following set of nodes:

$$i \prec n+i \prec 2n+i \prec 3n+i \qquad \forall i \in \mathcal{P}^+ \tag{6.1}$$

where $i \prec j$ denotes the precedence relation between nodes $i$ and $j$, i.e., the constraint indicating that $i$ must be visited before $j$ if both $i$ and $j$ are served by the same AV route.

Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be a directed graph with a node set $\mathcal{N} = \mathcal{P} \cup \mathcal{D} \cup \{v_s, v_t\}$ containing all pickup and drop-off nodes together with a source and a sink node (both representing the designated depot) and an edge set $\mathcal{A} = \{(i,j) : i, j \in \mathcal{N}, i \neq j\}$ consisting of all possible edges as a first approximation. A time window $[a_i, b_i]$ and a service duration $\zeta_i$ are then associated with each node $i \in \mathcal{P} \cup \mathcal{D}$. No time windows are associated with $v_s$ and $v_t$ as it is assumed that the AVs may start and end their routes at any time of the day. Additionally, a ride-duration limit $L_i$ is associated with each node $i \in \mathcal{P}$. Finally, a travel time $\tau_{(i,j)}$, a distance $\varsigma_{(i,j)}$, and a cost $c_{(i,j)}$ are associated with each edge $(i,j) \in \mathcal{A}$, and $\delta^+(i)$ and $\delta^-(i)$ denote the sets of all outgoing and incoming edges of node $i$ respectively.

### 6.2.2 The MIP Model for the CTSPAV

This section revisits the MIP model for the CTSPAV. The MIP is defined by (6.2)–(6.12): It formalizes the CTSPAV and is defined on the graph $\mathcal{G}$ and *the set* $\Omega$ *of all feasible mini*

*routes*. The MIP formulation uses two sets of binary variables: Variable $X_r$ indicates whether mini route $r \in \Omega$ is selected and variable $Y_{(i,j)}$ indicates whether edge $(i,j) \in \mathcal{A}$ is used, i.e., whether node $j$ should be visited immediately after node $i$ by an AV route in the optimal solution. Additionally, the model uses a continuous variable $T_i$ that represents the start of service time at node $i \in \mathcal{P} \cup \mathcal{D}$.

$$\min \quad \sum_{e \in \mathcal{A}} c_e Y_e \tag{6.2}$$

$$\text{s.t.} \quad \sum_{r \in \Omega : i \in r} X_r = 1 \qquad \forall i \in \mathcal{P} \tag{6.3}$$

$$\sum_{r \in \Omega : e \in r} X_r - Y_e \leq 0 \qquad \forall e \in \mathcal{A} \setminus \{\delta^+(v_s) \cup \delta^-(v_t)\} \tag{6.4}$$

$$\sum_{e \in \delta^+(i)} Y_e = 1 \qquad \forall i \in \mathcal{P} \cup \mathcal{D} \tag{6.5}$$

$$\sum_{e \in \delta^-(i)} Y_e = 1 \qquad \forall i \in \mathcal{P} \cup \mathcal{D} \tag{6.6}$$

$$T_i + \zeta_i + \tau_{(i,j)} \leq T_j + M_{(i,j)}(1 - Y_{(i,j)}) \qquad \forall i,j \in \mathcal{P} \cup \mathcal{D} \tag{6.7}$$

$$T_i + \zeta_i + \tau_{(i,j)} \geq T_j - \bar{M}_{(i,j)}(1 - Y_{(i,j)}) \qquad \forall i \in \mathcal{P} \cup \mathcal{D}, \forall j \in \mathcal{D} \tag{6.8}$$

$$T_{i+n} - (T_i + \zeta_i) \leq L_i \qquad \forall i \in \mathcal{P} \tag{6.9}$$

$$a_i \leq T_i \leq b_i \qquad \forall i \in \mathcal{P} \cup \mathcal{D} \tag{6.10}$$

$$X_r \in \{0,1\} \qquad \forall r \in \Omega \tag{6.11}$$

$$Y_e \in \{0,1\} \qquad \forall e \in \mathcal{A} \tag{6.12}$$

The objective function (6.2) minimizes the total cost of all selected edges. Contraints (6.3) ensure each trip is served by exactly one mini route, while constraints (6.4) select edges belonging to selected mini routes. Constraints (6.5) and (6.6) simultaneously ensure each pickup and drop-off node is visited exactly once while conserving the flow through each. Constraints (6.7) and (6.8) ensure the start of service times at the tail and the head of every selected edge is compatible with the travel time along the edge using large constants $M_{(i,j)}$ and $\bar{M}_{(i,j)}$. Finally, constraints (6.9) and (6.10) describe the ride-duration limit of every trip and the time-window constraint of every pickup and drop-off node respectively.

Note that constraints (6.7) and (6.8) are generalizations of the popular Miller-Tucker-Zemlin (MTZ) SECs for the TSP (Miller et al. 1960). They utilize the following big-$M$

constants to enforce the underlying constraints on the selected edges:

$$M_{(i,j)} = \max\{0, b_i + \zeta_i + \tau_{(i,j)} - a_j\} \qquad \forall i, j \in \mathcal{P} \cup \mathcal{D} \tag{6.13}$$

$$\bar{M}_{(i,j)} = \max\{0, b_j - a_i - \zeta_i - \tau_{(i,j)}\} \qquad \forall i \in \mathcal{P} \cup \mathcal{D}, \forall j \in \mathcal{D} \tag{6.14}$$

The model adopts a lexicographic objective whose primary objective is to minimize the number of vehicles used and whose secondary objective is to minimize the total travel distance. This lexicographic ordering is accomplished by weighting the sub-objectives: An identical, large fixed cost and a variable cost that is proportional to the route total distance are assigned to each AV route. The edge costs are defined as follows to accomplish this goal:

$$c_e = \begin{cases} \varsigma_e + 100\hat{\varsigma}_{\text{max}} & \forall e \in \delta^+(v_s) \\ \varsigma_e & \text{otherwise} \end{cases} \tag{6.15}$$

where $\hat{\varsigma}_{\text{max}}$ is a constant equal to the length (total distance) of the longest AV route. Letting $\mathcal{R}$ denote the set of all feasible AV routes, $\hat{\varsigma}_{\text{max}}$ is given by:

$$\hat{\varsigma}_{\text{max}} = \max_{\rho \in \mathcal{R}} \sum_{(i,j) \in \rho} \varsigma_{(i,j)} \tag{6.16}$$

The CTSPAV model essentially solves a scheduling problem that selects and assembles feasible mini routes to form longer, feasible AV routes to cover all trips while minimizing the total cost. The optimal AV routes are obtained by constructing paths beginning at $v_s$ and ending at $v_t$ from the selected edges, and their start and end times can be calculated using equations (1.7) and (1.9) respectively.

### 6.2.3 The Mini Route-Enumeration Algorithm

Since the MIP model is defined in terms of all feasible mini routes, this section describes the Mini Route-Enumeration Algorithm (MREA), a procedure for enumerating all the routes in $\Omega$ that is based on the REA proposed in Section 2.2.2. The set $\Omega$ can be partitioned into $\Omega = \Omega^+ \cup \Omega^-$, where $\Omega^+$ represents the set of all feasible inbound mini routes (which covers only inbound trips) and $\Omega^-$ represents the set of all feasible outbound mini routes (which covers only outbound trips). Without loss of generality, this section describes the procedure for enumerating the mini routes in $\Omega^+$.

The procedure is summarized in Algorithm 3. It requires as inputs the set $\mathcal{T}^+$ of all inbound trips and the vehicle capacity $K$. It begins by considering all feasible inbound mini routes for a vehicle capacity of 1 by adding the routes for all the direct trips from $\mathcal{T}^+$ to $\Omega^+$

**Algorithm 3** Mini Route-Enumeration Algorithm for $\Omega^+$

**Require:** $\mathcal{T}^+, K$

1: $\Omega^+ \leftarrow \emptyset$
2: **for** each $t_c^+ \in \mathcal{T}^+$ **do**
3: $\quad \Omega^+ \leftarrow \Omega^+ \cup \{o_c^+ \rightarrow d_c^+\}$
4: **for** $k = 2$ to $K$ **do**
5: $\quad \mathcal{Q}_k \leftarrow \{$all $k$-combinations of $\mathcal{T}^+\}$
6: $\quad$ **for** each $q \in \mathcal{Q}_k$ **do**
7: $\quad\quad \Omega_q^v \leftarrow \{$all valid mini routes of $q\}$
8: $\quad\quad$ **for** each $r^+ \in \Omega_q^v$ **do**
9: $\quad\quad\quad$ **if** $feasible(r^+)$ **then**
10: $\quad\quad\quad\quad \Omega^+ \leftarrow \Omega^+ \cup \{r^+\}$
11: **return** $\Omega^+$

(lines 2–3). It then enumerates the feasible routes for progressively increasing vehicle capacities by incrementing a parameter $k$ (which represents the considered vehicle capacity) from 2 to $K$ (line 4). For each $k$, the procedure first enumerates all $k$-combinations of trips from $\mathcal{T}^+$ (line 5). Let $\mathcal{Q}_k$ represent the set of all $k$-trip combinations. It then enumerates all the valid mini routes for each trip combination $q \in \mathcal{Q}_k$. Let $\Omega_q^v$ be this set of routes for a trip combination $q$. The procedure finally checks the feasibility of each route in $\Omega_q^v$ (using the $feasible$ function described in Section 1.1.3) and adds the ones that are feasible to $\Omega^+$ (lines 8–10).

In summary, the enumeration procedure considers all trip combinations of size $k \leq K$ (of which there are $O(n^K)$ combinations when the vehicle capacity $K$ is fixed). It therefore performs $O(n^K)$ iterations, within each which a trip combination is considered. For each $k$-combination, it enumerates $(k!)^2$ valid route permutations ($k!$ pickup node permutations followed by $k!$ drop-off node permutations for each pickup permutation) and checks the feasibility of each. Therefore, *for each iteration*, $O((K!)^2)$ mini routes are enumerated. Additionally, since this work implements the route-feasibility check by Gschwind and Irnich (2015) which has an $O(K^2)$ time complexity, the enumeration procedure *for each iteration* has a time complexity of $O(K^2(K!)^2)$. Therefore, when the vehicle capacity $K$ is fixed (as is the case for this work), the space and time complexity of the entire MREA reduces to $O(n^K)$. Chapter 5 showed that capacities greater than 5 bring only marginal benefits in its case study, and this will also be confirmed later in this chapter.

The labeling procedure by Gschwind and Irnich (2015) makes it possible to perform feasibility checks when extending partial mini routes; it therefore permits a more efficient implementation of lines 7–10 of Algorithm 3. The set of feasible mini routes for any trip combination $q$ can be enumerated by performing a depth-first search which checks the fea-

sibility of each partial route as it is being extended and backtracks when an extension is infeasible. Furthermore, the independence of the search procedure for each trip combination $q \in \mathcal{Q}_k$ allows them to be performed in parallel.

### 6.2.4 Filtering of Graph $\mathcal{G}$

Graph $\mathcal{G}$ can be made more compact by only retaining edges that satisfy a priori route-feasibility constraints. This is done by pre-processing time-window, pairing, precedence, and ride-duration limit constraints on $\mathcal{A}$ to identify and eliminate edges that are infeasible, i.e., those that cannot belong to any feasible AV route. In this work, the set of infeasible edges is identified using a combination of rules proposed by Dumas et al. (1991) and Cordeau (2006). These rules are listed in Appendix C.1.

## 6.3 Valid Inequalities for the CTSPAV

The CTSPAV MIP is solved with a traditional branch-and-cut procedure that exploits a number of valid inequalities for the MIP formulation. The inequalities are valid for all nodes of the branch-and-bound tree, and the LP relaxation at each node incorporates all inequalities discovered up to that point. Numerous families of valid inequalities that have been proposed for the TSP (Dantzig et al. 1954, Grötschel and Padberg 1985, Padberg and Rinaldi 1991), the ATSP (Fischetti and Toth 1997), the PCATSP (Balas et al. 1995), the PDP (Ruland and Rodin 1997), the ATSPTW (Ascheuer et al. 2000, 2001), the VRPTW (Kohl et al. 1999, Bard et al. 2002, Kallehauge et al. 2007), the PDPTW (Ropke and Cordeau 2009), and the DARP (Cordeau 2006) are also valid for the CTSPAV as the CTSPAV is a specialization of the DARP. This work, however, considers only inequalities that specifically improve the lower bound of the vehicle count (the primary objective). This is because extensive computational experiments from Chapter 5 showed that the LP relaxation already provides a sufficiently strong lower bound for the secondary objective (total distance). This section describes the considered valid inequalities together with their corresponding separation heuristics when applicable. The following notation is used to simplify the exposition. For any set of edges $\mathcal{A}' \subseteq \mathcal{A}$, let $Y(\mathcal{A}') = \sum_{e \in \mathcal{A}'} Y_e$. For a set of nodes $S \subseteq \mathcal{N}$, let $\bar{S}$ denote its complement, i.e., $\bar{S} = \{i \in \mathcal{N} : i \notin S\}$. For any two node sets $S, T \subseteq \mathcal{N}$, let $(S, T) = \{(i, j) \in \mathcal{A} : i \in S, j \in T\}$. For brevity, $Y(S, T)$ is used to represent $Y((S, T))$. Finally, for node set $S \subseteq \mathcal{P} \cup \mathcal{D}$, let $\pi(S) = \{i \in \mathcal{P} : n + i \in S\}$ and $\sigma(S) = \{n + i \in \mathcal{D} : i \in S\}$ denote the sets of predecessors and successors of $S$ respectively.

### 6.3.1 Rounded Vehicle-Count Inequalities

Suppose that a (fractional) lower bound $\chi_{\text{LB}}$ is known for the vehicle count. The inequality

$$Y(\delta^+(v_s)) \geq \lceil \chi_{\text{LB}} \rceil \tag{6.17}$$

is a direct consequence of the integrality of the vehicle count. Such a lower bound can be obtained by selecting the best bound in the branch-and-bound algorithm. Let $Y_e^*$ denote the value of $Y_e$ in the LP solution of the best bound. A lower bound $\chi_{\text{BB}}$ can be obtained from

$$\chi_{\text{BB}} = \sum_{e \in \delta^+(v_s)} Y_e^* \tag{6.18}$$

and be used in place of $\chi_{\text{LB}}$ in (6.17).

#### 6.3.1.1 The Column-Generation Procedure for Deriving Vehicle-Count Lower Bounds

A stronger lower bound may be obtained from a column-generation procedure that solves the CTSPAV as a DARP. In Chapter 5, we discovered that a column-generation procedure which resembles that used by Gschwind and Irnich (2015) for solving the DARP is capable of producing strong lower bounds for the vehicle count of the CTSPAV when it is paired with an appropriate objective function. This work leverages the procedure—by creating and then solving a redundant DARP model—to strengthen the vehicle-count lower bound of the CTSPAV MIP.

The DARP column-generation procedure from Chapter 5 features a PSP that searches for AV routes with negative reduced costs to improve the objective function of a set-covering MP whose columns consist of the routes. More specifically, it utilizes a RMP which is the linear relaxation of the MP that is defined on a subset $\mathcal{R}' \subseteq \mathcal{R}$ of all feasible AV routes. The routes discovered by the PSP are progressively added to $\mathcal{R}'$ as the RMP and the PSP are solved iteratively. The column generation terminates when the PSP cannot identify any AV route with a negative reduced cost. At this stage, the objective value $z_{\text{RMP}}$ of the RMP converges to the optimal objective value $z^*$ of the linear relaxation of the original MP. In this work, the column-generation procedure is not used to obtain a solution to the CTSPAV per se; instead, it is just used to extract (potentially strong) lower bounds to the primary objective of the CTSPAV. The following describes the procedure for obtaining these lower bounds.

**The Restricted Master Problem** The RMP is given by the following set-covering formulation:

$$\min \quad z = \sum_{\rho \in \mathcal{R}'} X_\rho \tag{6.19}$$

$$\text{s.t.} \quad \sum_{\rho \in \mathcal{R}'} a_{i,\rho} X_\rho \geq 1 \qquad \forall i \in \mathcal{P} \tag{6.20}$$

$$X_\rho \geq 0 \qquad \forall \rho \in \mathcal{R}' \tag{6.21}$$

It is defined on a subset $\mathcal{R}' \subseteq \mathcal{R}$ of all feasible AV routes, and uses a variable $X_\rho$ to indicate whether AV route $\rho \in \mathcal{R}'$ is used in the optimal solution. Its objective function (6.19) minimizes the number $z$ of selected AV routes and is therefore identical to the primary objective of the CTSPAV. Constraints (6.20) ensure each pickup node is covered in the solution, and $a_{i,\rho}$ is a constant that indicates the number of times node $i$ is visited by route $\rho$.

**The Pricing Subproblem** The PSP searches for AV routes with negative reduced costs to be added to $\mathcal{R}'$. It uses $\{\mu_i : i \in \mathcal{P}\}$, the set of optimal duals of constraints (6.20), to compute the reduced costs of the undiscovered routes. The reduced cost of a route $\rho$ is given by:

$$\bar{c}_\rho = 1 - \sum_{i \in \mathcal{P}} a_{i,\rho} \mu_i. \tag{6.22}$$

To find these routes, a graph $\mathcal{G}$ identical to that defined in Section 6.2 is first constructed. A reduced cost $\bar{c}_{(i,j)}$ is then associated with each edge $(i,j) \in \mathcal{A}$, and it is defined as follows so that the total cost of any path in $\mathcal{G}$ from $v_s$ to $v_t$ is equivalent to (6.22):

$$\bar{c}_{(i,j)} = \begin{cases} 1 & \forall (i,j) \in \delta^+(v_s) \\ -\mu_i & \forall i \in \mathcal{P}, \forall j \in \mathcal{N} \\ 0 & \forall i \in \mathcal{D}, \forall j \in \mathcal{N}. \end{cases} \tag{6.23}$$

Obtaining a solution to the PSP is then a matter of finding a feasible AV route, i.e., a path from $v_s$ to $v_t$ that satisfies the time-window, capacity, pairing, precedence, and ride-duration limit constraints, with a negative reduced cost. The PSP can be solved by first finding the least-cost feasible path from $v_s$ to $v_t$ and then adding it to $\mathcal{R}'$ if the cost is negative. This approach makes the problem an ESPPRC which can be solved by the label-setting dynamic program proposed by Gschwind and Irnich (2015). The necessity of ensuring elementarity of the path (to ensure its feasibility), however, makes the problem especially hard to

solve (Dror 1994). Since we are only interested in deriving lower bounds to the vehicle count from this procedure and not in discovering the AV routes per se, the elementarity requirement can be relaxed to admit a pseudo-polynomial solution from the label-setting algorithm. While the relaxation, in theory, may cause $z_{\text{RMP}}$ to converge to a weaker primal bound as the PSP would admit a larger set of routes $\mathcal{R}'' \supseteq \mathcal{R}'$, other works that have adopted a similar strategy (e.g., Ropke and Cordeau (2009) and Gschwind and Irnich (2015)) have discovered that the lower bound is only slightly weaker in practice.

**Extracting a Lower Bound to the Vehicle Count from the PSP** As mentioned earlier, $z_{\text{RMP}}$ converges to $z^*$ and therefore becomes a valid lower bound to the vehicle count of the CTSPAV when the PSP is unable to discover any AV route with a negative reduced cost. However, reaching this point in the procedure typically requires many column-generation iterations and thus a long computation time. Prior to it, $z_{\text{RMP}}$ only represents an upper bound to $z^*$ and therefore it cannot be used to bound the vehicle count. Fortunately, the identical unit cost of each AV route in the RMP allows for the derivation of a lower bound to $z^*$ using the method proposed by Farley (1990). The Farley bound after the $k^{\text{th}}$ column-generation iteration is given by:

$$z^k_{\text{Farley}} = \frac{z_{\text{RMP}}}{1 - \bar{c}^k_\rho} \tag{6.24}$$

where $\bar{c}^k_\rho$ represents the smallest route-reduced cost discovered by the PSP after the $k^{\text{th}}$ iteration. As the value of $z^k_{\text{Farley}}$ tends to fluctuate between iterations, a monotonically non-decreasing lower bound to $z^*$ can be obtained with the following equation:

$$z^k_{\text{Farley}} = \max\left\{\frac{z_{\text{RMP}}}{1 - \bar{c}^k_\rho}, z^{k-1}_{\text{Farley}}\right\} \tag{6.25}$$

As $z^k_{\text{Farley}}$ is a lower bound to $z^*$, it is also a valid lower bound to the vehicle count of the CTSPAV. Therefore, $\chi_{\text{LB}}$ for cut (6.17) may be defined as follows:

$$\chi_{\text{LB}} = \max\left\{\chi_{\text{BB}}, z^k_{\text{Farley}}\right\} \tag{6.26}$$

Since $z^k_{\text{Farley}}$ as defined in (6.25) is monotonically non-decreasing and improves with the number of column-generation iterations, it is practical to dedicate a single computational thread for executing this column-generation procedure and use the remaining thread(s) for solving the CTSPAV MIP in parallel. The CTSPAV MIP may then check for the most up-to-date value of $z^k_{\text{Farley}}$ from the column-generation thread after evaluating the LP relaxation of each tree node and introduce cut (6.17) when there is an improvement to the rounded vehicle-count lower bound.

### 6.3.2 Two-Path Inequalities

The two-path inequality was originally conceived by Kohl et al. (1999) for the VRPTW. It has been shown to be particularly effective at strengthening the lower bound for the vehicle count of the VRPTW (Bard et al. 2002) and the PDPTW (Ropke and Cordeau 2009) when vehicle-count minimization is (part of) the objective function. For a set of nodes $S \subseteq \mathcal{P} \cup \mathcal{D}$, let $\kappa(S)$ denote the *minimum* number of vehicles needed to serve $S$, i.e., the minimum number of vehicles needed to serve all the nodes in $S$ while satisfying all route-feasibility constraints. The following two-path inequality,

$$Y(S, \bar{S}) \geq 2 \qquad \forall S \subseteq \mathcal{P} \cup \mathcal{D}, \kappa(S) > 1 \tag{6.27}$$

is valid when it is known that a single vehicle cannot feasibly serve all the nodes of set $S$, i.e., when $\kappa(S) > 1$. Inequality (6.27) has a form that is similar to the cutset inequality:

$$Y(S, \bar{S}) \geq 1 \qquad \forall S \subseteq \mathcal{P} \cup \mathcal{D}, |S| \geq 2 \tag{6.28}$$

which, in turn, is equivalent to the DFJ SEC (Dantzig et al. 1954):

$$Y(S, S) \leq |S| - 1 \qquad \forall S \subseteq \mathcal{P} \cup \mathcal{D}, |S| \geq 2 \tag{6.29}$$

Inequalities (6.28) or (6.29) are typically used to eliminate subtours in the TSP and the ATSP (the subtours manifest themselves as cycles when two separate nodes are used to represent the depot, as is done in this work). For instance, (6.28) does so by requiring at least a unit of flow to emanate from any set $S$ with two or more nodes. The two-path inequality (6.27) can therefore be seen as a strengthened SEC as it requires at least two units of flow to emanate from any set $S$. However, its validity is contingent upon a stronger condition, i.e., $\kappa(S) > 1$.

For the CTSPAV, a method similar to that proposed by Ropke and Cordeau (2009) may be used to determine if $\kappa(S) > 1$ for any $S$. It essentially requires one to determine if there exists a *feasible* path that first visits all the nodes in $\pi(S) \setminus S$, followed by all the nodes in $S$, and then all the nodes in $\sigma(S) \setminus S$. If such a path does not exist, then $\kappa(S) > 1$. The task of determining the existence of this path can be accomplished by first constructing a three-layered graph $\mathcal{G}_S = (\mathcal{N}_S, \mathcal{A}_S)$ with a node set $\mathcal{N}_S = \pi(S) \cup S \cup \sigma(S) \cup \{v_s, v_t\}$ and an initially empty edge set $\mathcal{A}_S$. The nodes from $\mathcal{N}_S \setminus \{v_s, v_t\}$ are then grouped into three layers, the first consisting of $\pi(S) \setminus S$, the second consisting of $S$, and the third containing $\sigma(S) \setminus S$. The following sets of edges are then introduced into $\mathcal{A}_S$:

- $\{(v_s, v_t)\}$

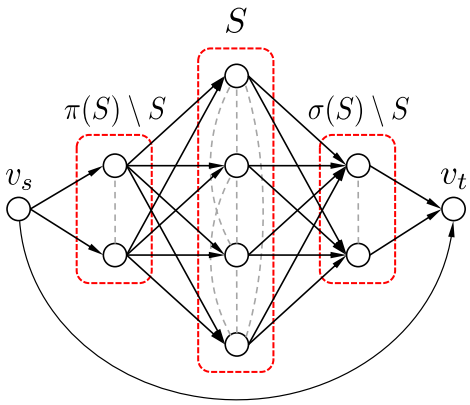



Figure 6.2: Graph $\mathcal{G}_S$ (Each Dotted Line Represents a Pair of Bidirectional Edges).

- $(\{v_s\}, \pi(S) \setminus S) \cap \mathcal{A}$
- $(\pi(S) \setminus S, S) \cap \mathcal{A}$
- $(S, \sigma(S) \setminus S) \cap \mathcal{A}$
- $(\sigma(S) \setminus S, \{v_t\}) \cap \mathcal{A}$
- $(\pi(S) \setminus S, \pi(S) \setminus S) \cap \mathcal{A}$
- $(S, S) \cap \mathcal{A}$
- $(\sigma(S) \setminus S, \sigma(S) \setminus S) \cap \mathcal{A}$

with $\mathcal{A}$ denoting the set of feasible edges of the graph $\mathcal{G}$ described in Section 6.2 (after it has been filtered). Figure 6.2 provides a sketch of $\mathcal{G}_S$. The following sets of edges are introduced into $\mathcal{A}_S$ should either $\pi(S) \setminus S$ or $\sigma(S) \setminus S$ be empty:

- If $\pi(S) \setminus S = \emptyset$, introduce $(\{v_s\}, S) \cap \mathcal{A}$
- If $\sigma(S) \setminus S = \emptyset$, introduce $(S, \{v_t\}) \cap \mathcal{A}$

One now needs to determine if there exists a feasible path from $v_s$ to $v_t$ that visits every node of $\mathcal{G}_S$. This problem can be treated as an ESPPRC, whereby an edge cost of $-1$ is first assigned to all the edges leaving the pickup nodes of $\mathcal{N}_S$ (i.e., the edges in $(\pi(S), \mathcal{N}_S \setminus \pi(S))$). A feasible path from $v_s$ to $v_t$ that visits every node of $\mathcal{G}_S$ then exists if and only if the least-cost *elementary* path from $v_s$ to $v_t$ has a total cost of $-|\pi(S)|$. While this ESPPRC is well known to be NP-hard (Dror 1994), it can be solved efficiently using the label-setting algorithm by Gschwind and Irnich (2015) when $S$ is small. Therefore, one just needs to solve the ESPPRC and check the total cost of the resulting elementary path. If the cost is greater than $-|\pi(S)|$, then the nodes of $S$ cannot be feasibly served by a single vehicle, $\kappa(S) > 1$, and (6.27) becomes a valid inequality.

#### 6.3.2.1 Separation Heuristic

The separation heuristic for the two-path inequalities first identifies sets of nodes $S$ for which $\kappa(S) > 1$. As the two-path inequality is essentially a strengthened SEC, the heuristic first identifies sets of nodes that form subtours (cycles) in the LP relaxation at each tree node. Let $Y_e^*$ denote the value of $Y_e$ from the LP solution. For every subtour $S$ considered, the heuristic then checks if $\sum_{e \in (S,\bar{S})} Y_e^* < 2$ and then if $\kappa(S) > 1$. Satisfaction of these two conditions indicates that the two-path inequality is valid for $S$ and that it is violated in the LP relaxation. The heuristic therefore adds the two-path inequality for $S$ to eliminate the subtour from subsequent LP solutions.

To identify subtours from an LP relaxation, the heuristic by Drexl (2013) is used. The heuristic was proposed as a cheaper yet effective alternative for identifying violated SECs to the exact method proposed by Gomory and Hu (1961), as it has an $O(n^2)$ complexity compared to the $O(n^4)$ complexity of the latter. For any LP solution, a support graph, $\mathcal{G}_{\text{sp}} = (\mathcal{N}_{\text{sp}}, \mathcal{A}_{\text{sp}})$, is first constructed with nodes $\mathcal{N}_{\text{sp}} = \mathcal{N}$ and edges $\mathcal{A}_{\text{sp}} = \{e \in \mathcal{A} : Y_e^* > 0\}$. All strongly-connected components (SCCs) of $\mathcal{G}_{\text{sp}}$ are then identified, where an SCC of a graph is its subgraph with more than one node whereby there exists a path between all pairs of its nodes. The rationale behind the identification of SCCs is that each constitutes a subtour (the nodes of the SCC form a cycle(s) as every node is reachable from another). In practice, all SCCs of $\mathcal{G}_{\text{sp}}$ can be computed using the algorithm by Tarjan (1972) which has a time complexity of $O(|\mathcal{N}_{\text{sp}}| + |\mathcal{A}_{\text{sp}}|)$. Let $\mathcal{S}_{\text{sp}}$ denote the set of all SCCs of $\mathcal{G}_{\text{sp}}$, and for each SCC $c \in \mathcal{S}_{\text{sp}}$, let $S_c$ denote its set of nodes. For every $c \in \mathcal{S}_{\text{sp}}$, the heuristic then checks if the total flow emanating from $S_c$ is less than 2, i.e., if $\sum_{e \in (S_c,\bar{S}_c)} Y_e^* < 2$. If this condition is satisfied, the heuristic then determines if $\kappa(S_c) > 1$ using the procedure described earlier. Finally, the two-path cut $Y(S_c, \bar{S}_c) \geq 2$ is introduced into the CTSPAV MIP if $\kappa(S_c) > 1$.

Due to the expensive nature of the procedure for determining if $\kappa(S_c) > 1$, results of the procedure for every set $S_c$ are stored in a hash table, and the hash table is examined first before the procedure is performed on any set $S$ to ensure that the same calculations are not repeated. Furthermore, the part of the procedure which solves the ESPPRC on graph $\mathcal{G}_S$ can also be made more efficient. Instead of directly applying the label-setting algorithm of Gschwind and Irnich (2015) which proposes keeping track of all visited pickup nodes and preventing path extensions to the already visited nodes to ensure elementarity, the procedure proposed by Boland et al. (2006) can be used. The latter entails iteratively solving a sequence of relaxed SPPRCs, whereby the elementarity requirement is completely relaxed in the very beginning. A repeated node from the solution of the relaxed problem is then selected and added to a set $\mathcal{U}$, after which the problem is solved again, this time with an ad-

ditional restriction that the nodes in $\mathcal{U}$ can only be visited once. The procedure is repeated with $\mathcal{U}$ being progressively enlarged until an elementary path is discovered. The rationale behind this procedure is that solving the sequence of relaxed SPPRCs is usually less expensive than solving a single ESPPRC in practice, as often times the former discovers an elementary path without having to include all the pickup nodes in $\mathcal{U}$. Desaulniers et al. (2008) proposed adding only the first repeated node from the solution of the relaxed problem to $\mathcal{U}$ after each iteration, and our preliminary evaluations showed that this approach works very well in practice.

### 6.3.3 Predecessor and Successor Inequalities

Predecessor and successor inequalities were first introduced by Balas et al. (1995) for the PCATSP. The predecessor inequality ($\pi$-inequality) is given by:

$$Y(S \setminus \pi(S), \bar{S} \setminus \pi(S)) \geq 1 \qquad \forall S \subseteq \mathcal{P} \cup \mathcal{D}, |S| \geq 2 \tag{6.30}$$

and the successor inequality ($\sigma$-inequality) is given by:

$$Y(\bar{S} \setminus \sigma(S), S \setminus \sigma(S)) \geq 1 \qquad \forall S \subseteq \mathcal{P} \cup \mathcal{D}, |S| \geq 2 \tag{6.31}$$

These inequalities are essentially lifted versions of the cutset inequality (6.28). They are also valid for the CTSPAV as it adopts precedence constraints that are similar to the PCATSP.

#### 6.3.3.1 Separation Heuristic

The heuristic utilized to separate $\pi$- and $\sigma$-inequalities is very similar to that described in Section 6.3.2.1 for the two-path inequality. At each tree node, values of $Y_e^*$ are first used to construct a support graph $\mathcal{G}_{\text{sp}}$, after which $\mathcal{S}_{\text{sp}}$ which represents the set of all SCCs of $\mathcal{G}_{\text{sp}}$ are identified. For each $c \in \mathcal{S}_{\text{sp}}$, the heuristic then checks if either inequalities (6.30) or (6.31) are violated for $S_c$, i.e., if either $\sum_{e \in (S_c \setminus \pi(S_c), \bar{S}_c \setminus \pi(S_c))} Y_e^* < 1$ or $\sum_{e \in (\bar{S}_c \setminus \sigma(S_c), S_c \setminus \sigma(S_c))} Y_e^* < 1$. Finally, corresponding $\pi$- or $\sigma$-inequalities are introduced into the CTSPAV MIP for each violation.

### 6.3.4 Lifted MTZ Inequalities

The lifted MTZ inequality was initially proposed by Desrochers and Laporte (1991) for the VRPTW. They were intended to strengthen MTZ constraints that are similar to (6.7) and

(6.8) which are well known to cause weak LP relaxations (Langevin et al. 1990, Gouveia and Pires 1999). The MTZ constraints for an edge $(i, j)$ is strengthened by taking into consideration the flow along the opposite edge $(j, i)$ combined with the fact that only one of the edges may have positive flow in a feasible integer solution. The lifted versions constraints (6.7) and (6.8) are given by (6.32) and (6.33) respectively.

$$T_i + \zeta_i + \tau_{(i,j)} \leq T_j + M_{(i,j)}(1 - Y_{(i,j)}) - \alpha_{(j,i)} Y_{(j,i)} \qquad \forall i, j \in \mathcal{P} \cup \mathcal{D} \tag{6.32}$$

$$T_i + \zeta_i + \tau_{(i,j)} \geq T_j - \bar{M}_{(i,j)}(1 - Y_{(i,j)}) - \beta_{(j,i)} Y_{(j,i)} \qquad \forall i \in \mathcal{P} \cup \mathcal{D}, \forall j \in \mathcal{D} \tag{6.33}$$

To correctly lift the constraints using this technique, the coefficients of the flow variable of the opposite edge, $\alpha_{(j,i)}$ and $\beta_{(j,i)}$, are assigned values that are as large as possible while ensuring that inequalities (6.32) and (6.33) are still valid for any feasible integer solution. Desrochers and Laporte (1991) proposed coefficient values for the VRPTW that ensure the earliest start of service times for every node. As serving pickup nodes as early as possible may not be desirable for the CTSPAV (as doing so lengthens the ride duration of the picked-up rider and thus increases the likelihood of exceeding her ride-duration limit), the coefficients are adjusted to (6.34) and (6.35) for the CTSPAV.

$$\alpha_{(j,i)} = \begin{cases} M_{(i,j)} - \zeta_i - \tau_{(i,j)} - \zeta_j - \tau_{(j,i)} & \text{if } i \in \mathcal{D} \\ M_{(i,j)} - \zeta_i - \tau_{(i,j)} - b_i + a_j & \text{otherwise} \end{cases} \tag{6.34}$$

$$\beta_{(j,i)} = -\bar{M}_{(i,j)} - \zeta_i - \tau_{(i,j)} - \zeta_j - \tau_{(j,i)} \tag{6.35}$$

The validity of the lifted constraints can be verified by first substituting (6.34) and (6.35) into (6.32) and (6.33) respectively, and then setting the flows along both edges $(i, j)$ and $(j, i)$ to zero or setting the flow along either edge to one. First, setting $Y_{(i,j)}$ and $Y_{(j,i)}$ to zero just disables constraints (6.32) and (6.33) for both edges. Next, setting $Y_{(i,j)} = 1$ and $Y_{(j,i)} = 0$ produces the following constraints,

$$T_i + \zeta_i + \tau_{(i,j)} \leq T_j \qquad \text{if } i, j \in \mathcal{P} \cup \mathcal{D} \tag{6.36}$$

$$T_i + \zeta_i + \tau_{(i,j)} \geq T_j \qquad \text{if } i \in \mathcal{P} \cup \mathcal{D}, j \in \mathcal{D} \tag{6.37}$$

which simply enforce the increasing service time requirement along edge $(i, j)$. Finally,

setting $Y_{(i,j)} = 0$ and $Y_{(j,i)} = 1$ results in the following set of constraints:

$$T_j + \zeta_j + \tau_{(j,i)} \geq T_i \qquad \text{if } j \in \mathcal{P} \cup \mathcal{D}, i \in \mathcal{D} \tag{6.38}$$

$$T_i - T_j \leq b_i - a_j \qquad \text{if } j \in \mathcal{P} \cup \mathcal{D}, i \in \mathcal{P} \tag{6.39}$$

$$T_j + \zeta_j + \tau_{(j,i)} \leq T_i \qquad \text{if } j \in \mathcal{D}, i \in \mathcal{P} \cup \mathcal{D} \tag{6.40}$$

Constraints (6.38) and (6.40) simply enforce increasing service times along edge $(j, i)$, while (6.39) is obviously a valid inequality if edge $(j, i)$ is selected.

### 6.3.5 Lifted Time-Bound Inequalities

The lifted time-bound inequalities were also proposed by Desrochers and Laporte (1991) to strengthen the time-window constraints of the VRPTW. Inequalities (6.41) and (6.42) strengthen the time-window constraints of node $i$ by taking into consideration the temporal requirements along the node's incoming and outgoing edges with positive flow.

$$T_i \geq a_i + \sum_{(j,i)\in\delta^-(i)} \max\{0, a_j - a_i + \zeta_j + \tau_{(j,i)}\} Y_{(j,i)} \qquad \forall i \in \mathcal{P} \cup \mathcal{D} \tag{6.41}$$

$$T_i \leq b_i - \sum_{(i,j)\in\delta^+(i)} \max\{0, b_i - b_j + \zeta_i + \tau_{(i,j)}\} Y_{(i,j)} \qquad \forall i \in \mathcal{P} \cup \mathcal{D} \tag{6.42}$$

## 6.4 Computational Results

This section presents the computational results of the branch-and-cut algorithm on problem instances derived from a real-world dataset of commute trips.

### 6.4.1 Algorithmic Settings

Three variants of the branch-and-cut algorithm are considered and contrasted in the evaluations; they are named CTSPAV$_{\text{Base}}$, CTSPAV$_{\text{SEC}}$, and CTSPAV$_{\text{Hybrid}}$. Each is differentiated by the types of valid inequalities included in its implementation. They are specified as follows:

- CTSPAV$_{\text{Base}}$ is the core algorithm and implements the simplest valid inequalities: lifted time bounds, lifted MTZ, and rounded vehicle count which uses only $\chi_{\text{BB}}$ as its lower bound;
- CTSPAV$_{\text{SEC}}$ is CTSPAV$_{\text{Base}}$ with the two-path, predecessor, and successor inequalities;
- CTSPAV$_{\text{Hybrid}}$ is CTSPAV$_{\text{Base}}$ with the DARP lower bound from Section 6.3.1.1.

The last variant also uses the interior-point, dual-stabilization method proposed by Rousseau et al. (2007) to accelerate the convergence of its DARP column-generation procedure. Furthermore, instead of only selecting the least-cost feasible path with a negative reduced cost in its PSP, all non-dominated paths resulting from the label-setting algorithm with negative reduced costs are added to $\mathcal{R}'$ to further accelerate convergence.

### 6.4.2 Construction of Problem Instances

Problem instances for the computational evaluations are derived from the Ann Arbor commute trip dataset. The performance evaluations utilize the trips made by the commuters living within Ann Arbor's city limits, the region bounded by highways US-23, M-14, and I-94. More specifically, the 2,200 commute trips from this region made on the busiest day of the month (Wednesday of week 2) were first selected and then partitioned into smaller problem instances using the spatial clustering algorithm described in Section 2.2.1. Trip sharing is then only considered intra-cluster with the largest parking structure arbitrarily designated as the depot for all clusters.

In addition to this, the following assumptions are made in order to define the time windows and ride-duration limit of each trip. Consistent with past works on the DARP (e.g. Jaw et al. (1986), Cordeau and Laporte (2003b), and Cordeau (2006)), each rider $i$ specifies a desired arrival time $at_i^+$ at the destination of her inbound trip and a desired departure time $dt_i^-$ at the origin of her outbound trip when making a request. The riders also tolerate a maximum shift of $\pm\Delta$ to the desired times. By treating the arrival and departure times to and from the parking structures as the desired times, an arrival-time upper bound at node $n+i$ of $b_{n+i} = at_i^+ + \Delta$ and a time window at node $2n+i$ of $[a_{2n+i}, b_{2n+i}] = [dt_i^- - \Delta, dt_i^- + \Delta]$ are defined for each $i \in \mathcal{P}^+$. Consequently, the time window at node $i$ is given by $[a_i, b_i] = [b_{n+i} - \zeta_i - L_i - 2\Delta, b_{n+i} - \zeta_i - L_i]$ and the arrival-time upper bound at node $3n+i$ is given by $b_{3n+i} = b_{2n+i} + \zeta_{2n+i} + L_{2n+i}$ for each $i \in \mathcal{P}^+$. Finally, consistent with Hunsaker and Savelsbergh (2002), the ride-duration limit of each trip is defined as an $R\%$ extension to the direct trip, i.e., $L_i = (1+R)\tau_{i,n+i}$ for each $i \in \mathcal{P}$.

A set of tight, medium, and large problem instances are constructed by varying parameter $N$ of the clustering algorithm together with $\Delta$ and $R$. The parameter combinations are carefully selected to highlight the performance differences of the three variants of the branch-and-cut algorithm considered. A vehicle capacity of $K = 4$ is used in all instances to represent the use of autonomous cars. Table 6.1 summarizes the parameters used together with the number of instances created when the clustering algorithm is applied on the set of 2,200 commuters.

Table 6.1: Parameters for Constructing Problem Instances

| Problem size | $N$ | $\Delta$ | $R$ | $K$ | Number of instances |
|---|---|---|---|---|---|
| Large | 100 | 10 mins | 0.50 | 4 | 22 |
| Medium | 75 | 10 mins | 0.50 | 4 | 30 |
| Tight | 100 | 5 mins | 0.25 | 4 | 22 |

Table 6.2: Average Vehicle-Count and Optimality Gaps of Every CTSPAV Variant for Every Problem Size

| CTSPAV variant | Average vehicle count gap | | | Average optimality gap | | |
|---|---|---|---|---|---|---|
| | Large | Medium | Tight | Large | Medium | Tight |
| Hybrid | 1.18 | 0.50 | 0.00 | 31.8% | 16.6% | 0.0% |
| SEC | 1.73 | 0.73 | 0.09 | 45.5% | 23.8% | 1.7% |
| Base | 2.50 | 1.67 | 0.14 | 68.0% | 59.0% | 3.2% |

### 6.4.3 Experimental Settings

All algorithms are implemented in C++. Parallelization of the MREA is handled with OpenMP, while the parallel execution of the column-generation procedure and the MIP of $\text{CTSPAV}_{\text{Hybrid}}$ is handled with the thread class from the C++11 standard library. All LPs and MIPs are solved with Gurobi 9.0.2, while graph algorithms from the Boost Graph Library (version 1.70.0) are used to calculate the SCCs of a graph and to implement the label-setting algorithm of Gschwind and Irnich (2015). Gurobi's callback feature is used to implement the bespoke cutting-plane separation and insertion, while the MIP solver is configured with its default parameters. For problem instance construction, Geocodio is used to geocode GPS coordinates of every address considered, after which GraphHopper's Directions API is used in conjunction with OpenStreetMap data to estimate the shortest path, travel time, and travel distance between any two nodes. Unless stated otherwise, every problem instance is solved on a compute cluster, each utilizing 4 cores of a 3.0 GHz Intel Xeon Gold 6154 processor and 16 GB of RAM. All four cores are used for the MREA. For $\text{CTSPAV}_{\text{Hybrid}}$, one core is dedicated for the column-generation procedure while the remaining three are used for solving the MIP. All four cores are used for solving the MIPs of $\text{CTSPAV}_{\text{SEC}}$ and $\text{CTSPAV}_{\text{Base}}$. Finally, a 2-hour time budget is allocated for solving every MIP.

### 6.4.4 Algorithm Performance Comparison

Table 6.2 first summarizes the average vehicle-count gaps and the average optimality gaps obtained for every problem size and every CTSPAV variant. Let $\chi_{\text{MIP}}$, $z_{\text{MIP}}$, and $z_{\text{BB}}$ de-

note the final vehicle count, the objective value of the best incumbent solution, and the best bound respectively. The vehicle-count gap is given by $\chi_{\text{MIP}} - \lceil \chi_{\text{LB}} \rceil$, while the optimality gap is given by $(z_{\text{MIP}} - z_{\text{BB}})/z_{\text{MIP}}$. The complete results of all the computational experiments are listed in Tables C.1–C.6 in Appendix C.2. Note that the route-enumeration time for every problem instance is consistently less than 60 seconds, which highlights the efficiency of the MREA.

The average optimality gaps for the large and medium instances appear to be relatively large. However, a closer examination of the vehicle-count gaps paints a different picture, as their values are relatively small across the board. In fact, the average vehicle-count gap for $\text{CTSPAV}_{\text{Hybrid}}$ is only a little above one for the large instances, and is less than one for the tighter instances. The gap values for $\text{CTSPAV}_{\text{Hybrid}}$ are also consistently smaller across the board than those of $\text{CTSPAV}_{\text{SEC}}$ which, in turn, are smaller than those of $\text{CTSPAV}_{\text{Base}}$. This observation provides the first evidence of the capability of $\text{CTSPAV}_{\text{Hybrid}}$'s column-generation procedure at producing very strong lower bounds for the primary objective; it also demonstrates the effectiveness of the combination of the two-path, successor, and predecessor inequalities at closing the vehicle-count gap (compared to an implementation that only adopts the basic inequalities). While the latter set of inequalities produces significant improvements in closing the primary gap, they are nevertheless outperformed by the rounded vehicle-count inequalities of $\text{CTSPAV}_{\text{Hybrid}}$.

Figure 6.3 provides a different perspective by summarizing the number of problem instances whose vehicle-count gaps are successfully closed within the 2-hour time limit for every CTSPAV variant. It also displays each count as a fraction of the total number of instances considered. For the large instances, $\text{CTSPAV}_{\text{Hybrid}}$ could only close the gap for three instances, while the other two variants could not for any of the problems from the set. This number improves for the medium instances, where $\text{CTSPAV}_{\text{Hybrid}}$ could now close the gap for 15 out of the 30 instances, while $\text{CTSPAV}_{\text{SEC}}$ could do the same for 9 of the instances. However, $\text{CTSPAV}_{\text{Base}}$ still cannot close the primary gap for any. Finally, for the tight instances, $\text{CTSPAV}_{\text{Hybrid}}$ finds the optimal solution for all of them, while $\text{CTSPAV}_{\text{SEC}}$ closes the primary gap for 90.9% of the instances and $\text{CTSPAV}_{\text{Base}}$ does the same for 86.4% of them. Regardless of the set of problem instances considered, the trend is clear: (1) The additional set of inequalities adopted by $\text{CTSPAV}_{\text{SEC}}$ allows it to successfully close the primary gap of more instances than $\text{CTSPAV}_{\text{Base}}$, and (2) $\text{CTSPAV}_{\text{Hybrid}}$ consistently outperforms the other two CTSPAV variants at closing the optimality gap. The latter observation provides yet another evidence of the efficacy of $\text{CTSPAV}_{\text{Hybrid}}$'s column-generation procedure at generating strong lower bounds for the primary objective.

Instead of aggregating the results from each problem set, Figures 6.4, 6.5, and 6.6 pro-

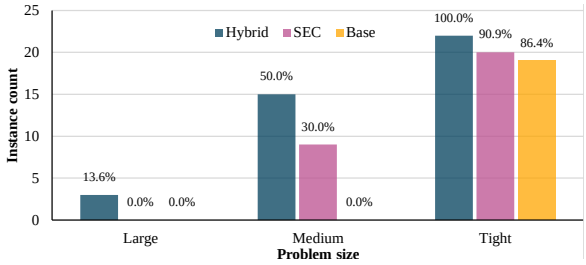



Figure 6.3: Number of Problem Instances Whereby Vehicle-Count Gap is Closed by Every CTSPAV Variant.

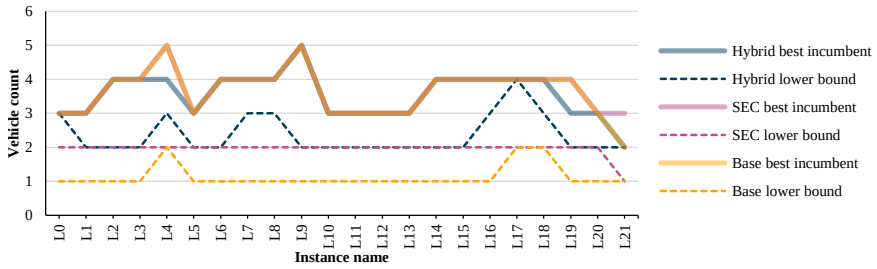



Figure 6.4: Best Incumbent Solution and Lower Bound for Vehicle Count of Every CTSPAV Variant for Every Large Problem Instance.

vide a more in-depth look at the final primary objective value and its corresponding lower bound for every problem instance from the large, medium, and tight sets respectively. For instance, Figure 6.4 shows the best incumbent solution and the lower bound for the vehicle count of every CTSPAV variant for every large problem instance. The figure reveals that, except for a few instances, all three variants produce identical final vehicle counts. The difference, however, lies in their lower bounds. The lower bounds of $\text{CTSPAV}_{\text{Hybrid}}$ dominate those of $\text{CTSPAV}_{\text{SEC}}$ in every instance. In turn, those of the latter dominate the lower bounds of $\text{CTSPAV}_{\text{Base}}$ in every instance as well. The same observation is carried over to Figure 6.5 which summarizes the primary gap of every instance from the medium set. While $\text{CTSPAV}_{\text{Hybrid}}$ and $\text{CTSPAV}_{\text{SEC}}$ produce identical lower bounds for more instances from this set, on the whole, the lower bounds of $\text{CTSPAV}_{\text{SEC}}$ are still dominated by those of $\text{CTSPAV}_{\text{Hybrid}}$. Similarly, they both dominate the lower bounds of $\text{CTSPAV}_{\text{Base}}$. Finally, Figure 6.6 summarizes the results of the tight instances and confirms the observations from the previous two figures. These observations lead to the following conclusion: Regardless of the size of the problem considered, there is a clear delineation between the strengths of the lower bound for the primary objective of the three CTSPAV variants. $\text{CTSPAV}_{\text{Hybrid}}$ dominates $\text{CTSPAV}_{\text{SEC}}$ which, in turn, dominates $\text{CTSPAV}_{\text{Base}}$. The relative strength of $\text{CTSPAV}_{\text{Hybrid}}$'s lower bound directly contributes to its ability to close or narrow the optimality gap of more problem instances than the other two variants.

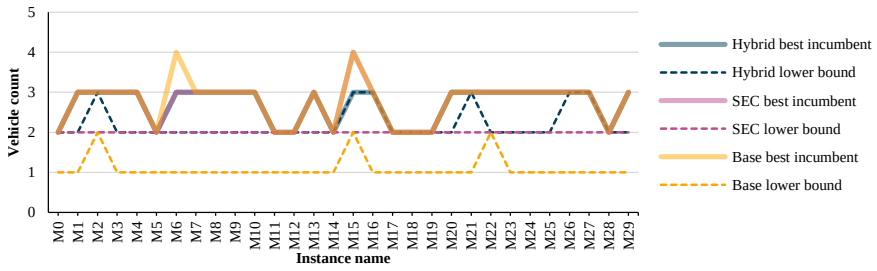



Figure 6.5: Best Incumbent Solution and Lower Bound for Vehicle Count of Every CTSPAV Variant for Every Medium Problem Instance.

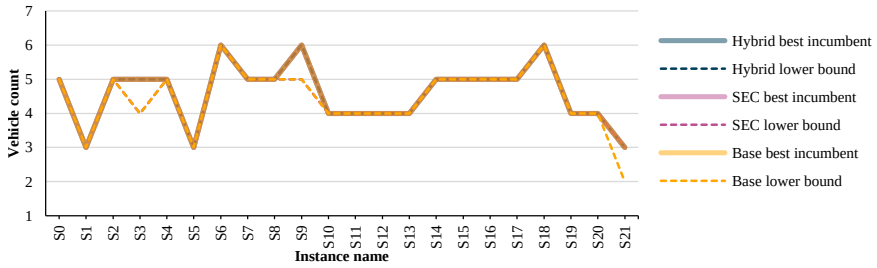



Figure 6.6: Best Incumbent Solution and Lower Bound for Vehicle Count of Every CTSPAV Variant for Every Tight Problem Instance.

### 6.4.5 Analysis of the Lower Bounds

Figure 6.7 presents a closer examination of the evolution of the best bound and the best incumbent objective value of every CTSPAV variant over time for a specific problem instance (instance L0). It also shows the progression of $z^k_{\text{Farley}}$ (after it has been scaled by the fixed cost $100\hat{\varsigma}_{\max}$) over time; the lower bound is obtained by rounding it up to the nearest multiple of $100\hat{\varsigma}_{\max}$. Since the MIP solver, with its default heuristics, is able to discover strong integer solutions fairly quickly for this formulation, the critical challenge lies in closing the optimality gap quickly. Unfortunately, the CTSPAV formulation uses big-$M$ constants in constraints (6.7) and (6.8) which produce weak LP relaxations.

The lifted MTZ and lifted time-bound inequalities provide only marginal improvements to the LP relaxation. While the rounded vehicle-count inequality has the capability of rectifying the issue, $\chi_{\text{BB}}$ rarely becomes fractional in practice, and thus the version of the inequality that only uses $\chi_{\text{BB}}$ rarely improves the vehicle-count lower bound. This explains why $\text{CTSPAV}_{\text{Base}}$ always produces the weakest lower bounds. Separation heuristics of the two-path, successor, and predecessor inequalities attempt to alleviate this situation by first searching for subtours that result from the flow of an LP relaxation, and then introducing the respective inequalities to remove these subtour flows from subsequent LP solutions. The experimental results of $\text{CTSPAV}_{\text{SEC}}$ demonstrate that these inequalities are indeed effective

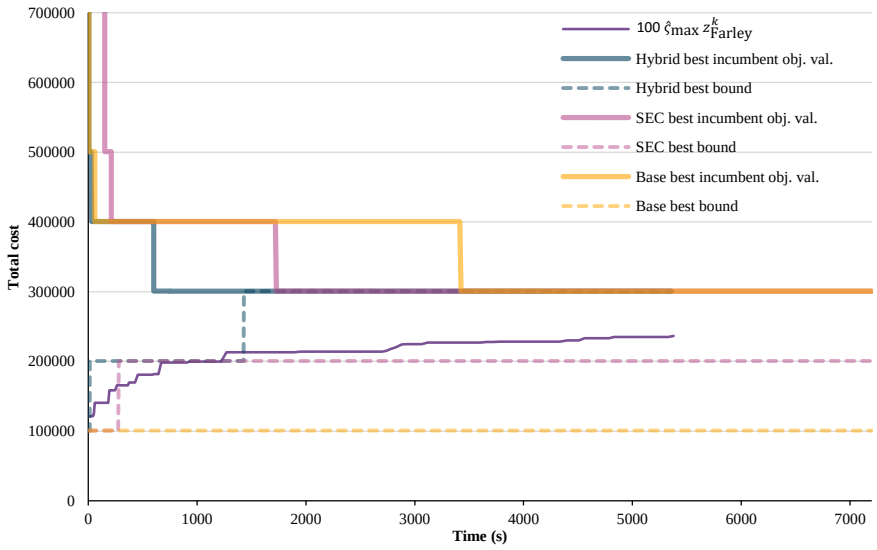



Figure 6.7: Evolution of Best Incumbent Objective Value and Best Bound of Every CTSP Variant for Problem Instance L0

at further strengthening the LP bound, however the results also show that their effect on the best bound tends to stagnate over time.

$\text{CTSPAV}_{\text{Hybrid}}$ attempts to circumvent the CTSPAV formulation's weak LP bound by dedicating a computational thread to solving the same problem using a (redundant) DARP formulation that focuses only on the primary objective. The Farley bound $z^k_{\text{Farley}}$ of the DARP relaxation provides a lower bound, and its scaled values in Figure 6.7 show that it progressively improves over time even after the best bounds of $\text{CTSPAV}_{\text{Base}}$ and $\text{CTSPAV}_{\text{SEC}}$ begin to stagnate. The ability of the column-generation procedure to produce relatively stronger lower bounds can be attributed to a few factors:

1. The RMP formulation does not utilize any big-$M$ constants.
2. The RMP uses only one set of binary variables ($X_\rho$), as opposed to two by the CTSPAV MIP ($X_r$ and $Y_e$). Therefore, fewer convex combinations of variables are allowed in its LP relaxation, which leads to stronger primal (and dual) lower bounds.
3. Ropke and Cordeau (2006) showed that the set-covering formulation actually implies several valid inequalities (precedence and strengthened precedence inequalities) that would otherwise need to be enforced explicitly in an edge-flow formulation.

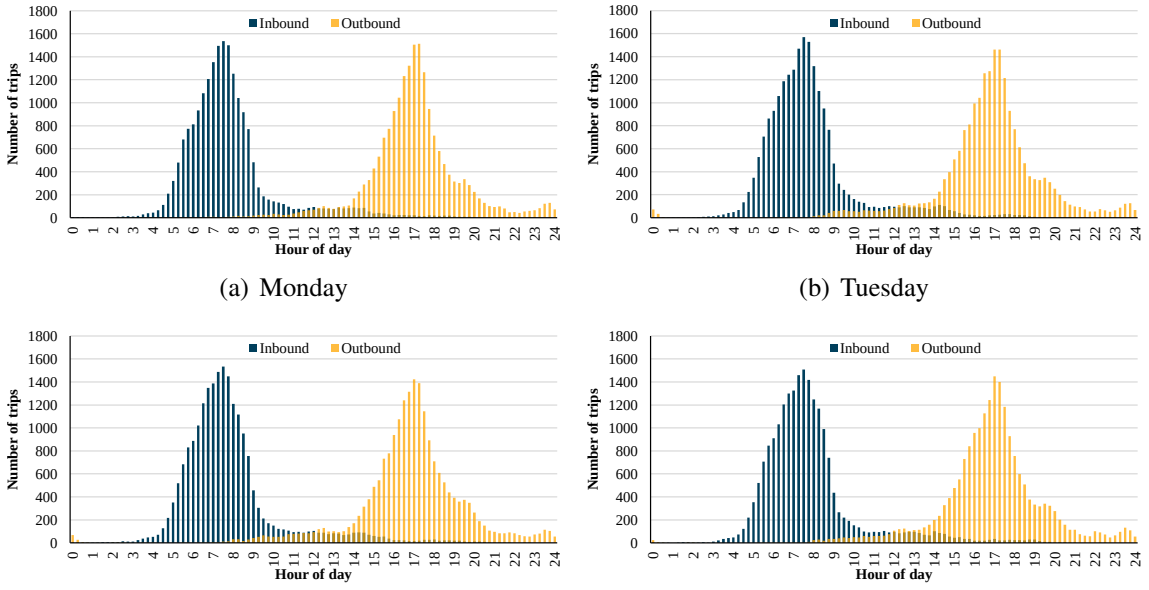
The approach of dedicating a single thread for executing the column-generation procedure also has a side benefit: It allows the branch-and-bound algorithm to freely explore more tree nodes without being encumbered by expensive separation heuristics. This is evident from a comparison of the number of explored nodes for several problem instances, for example, those of $\text{CTSPAV}_{\text{Hybrid}}$ and $\text{CTSPAV}_{\text{SEC}}$ for instances L1, L5, and L12 from

Tables C.1 and C.2. The comparison shows that the former is able to explore significantly more nodes, which could, in turn, lead to the discovery of better integer solutions. While $\text{CTSPAV}_{\text{Hybrid}}$ had one fewer thread for solving its MIP, it also did not have to execute any of the expensive separation heuristics of $\text{CTSPAV}_{\text{SEC}}$ which consequently resulted in a net gain in terms of the number of nodes it could explore.

### 6.4.6 Comparison with the CTSPAV Column-Generation Heuristic

It is useful to contrast the results from this work with those of the CTSPAV column-generation heuristic from Chapter 5. The heuristic does not exhaustively enumerate all the mini routes in $\Omega$. Instead, it uses a column-generation procedure consisting of a restricted master problem ($\text{RMP}_{\text{CTSPAV}}$)—the linear relaxation of MIP model (6.2)–(6.12) defined on only a subset $\Omega' \subseteq \Omega$ of the mini routes— and a pricing subproblem ($\text{PSP}_{\text{CTSPAV}}$) that searches for mini routes with negative reduced costs to augment $\Omega'$. The $\text{RMP}_{\text{CTSPAV}}$ and $\text{PSP}_{\text{CTSPAV}}$ are solved repeatedly until $\text{PSP}_{\text{CTSPAV}}$ is unable to find any mini route with a negative reduced cost. The heuristic then solves $\text{RMP}_{\text{CTSPAV}}$ as a MIP (that does not incorporate the valid inequalities considered in this work) to obtain a feasible integer solution. Since the heuristic only considers a subset of the feasible mini routes, *it is incapable of proving the optimality of its solution unless the solution of its* $\textit{RMP}_{\textit{CTSPAV}}$ *at convergence is integral* (which is never the case for the instances considered). Nevertheless, it is still instructive to compare its results against those of the exact $\text{CTSPAV}_{\text{Hybrid}}$ method to gauge the effectiveness of its column-generation procedure at identifying useful mini routes.

Tables C.7, C.8, and C.9 in Appendix C.2 give comprehensive results of the heuristic for every large, medium, and tight instance respectively. The results show that significantly fewer columns (mini routes) are considered by the heuristic. On average, it considers 66%, 62%, and 16% fewer columns for the large, medium, and tight instances respectively compared to $\text{CTSPAV}_{\text{Hybrid}}$. However, the final vehicle counts and total distances of the heuristic and $\text{CTSPAV}_{\text{Hybrid}}$ are very similar. In fact, the vehicle count results of the heuristic are identical to those of $\text{CTSPAV}_{\text{Hybrid}}$ in all except three instances: L19, M15, and S7. For these three instances, the counts of the heuristic are only greater than those of $\text{CTSPAV}_{\text{Hybrid}}$ by one vehicle. Moreover, the percentage difference in the total distance results are consistently less than 1.50% (on average, they differ by 0.01%). This similarity bodes very well for the heuristic; it highlights the effectiveness of its negative reduced cost criterion at identifying the subset of mini routes that is critical for producing strong integer solutions. It also indicates that the heuristic is more than sufficient for producing high-quality solutions, especially in applications where proving the optimality of the final solution is




(a) Monday (b) Tuesday

(c) Wednesday (d) Thursday

Figure 6.8: Commute Trip Demand Over 15-Minute Intervals on Week 2.

not of paramount importance. As mentioned earlier, the heuristic is incapable of closing the vehicle-count or optimality gap for any of the instances, so $\text{CTSPAV}_{\text{Hybrid}}$ remains the better candidate in applications where closing or narrowing the optimality gap is critical.

## 6.5 Case Study of Shared Commuting in Ann Arbor, Michigan

This section summarizes the results of a case study that applies the CTSPAV to optimize the commute trips from the Ann Arbor dataset. More specifically, it considers all trips (of commuters living inside and outside the city limits) from the first four weekdays (Monday–Thursday) of the busiest week of April 2017 (week 2). The parameters $N$, $\Delta$, and $R$ are set to 100, 10 minutes, and 50% respectively for this case study.[1] Its goal is to demonstrate the effectiveness of the CTSPAV at reducing vehicle usage and vehicle miles traveled, as well as to examine some of the real-world benefits and drawbacks of the AV ride-sharing platform.

Figure 6.8 provides an overview of the trip demand from the dataset and reports the number of ongoing trips for every 15-minute interval throughout the four days considered. The figure exhibits clear and consistent commuting patterns: The inbound demand peaks

[1] Part of the results for this case study is obtained by performing further analysis on the results from Chapter 5 which utilized the column-generation heuristic to solve the CTSPAV.

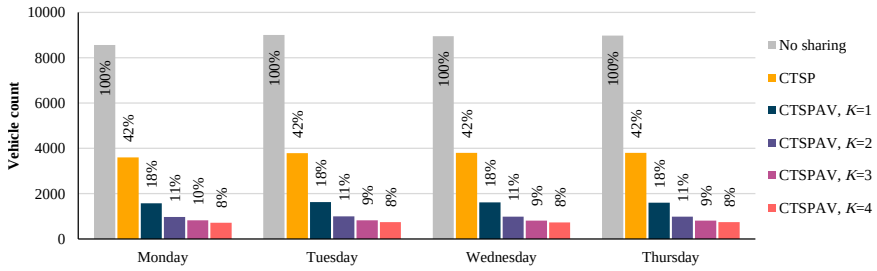



Figure 6.9: Total Number of Cars Used on Week 2.

between 7–8 am while the outbound demand peaks at around 5 pm every day. The highly consistent nature of the trip distribution highlights the opportunities for optimizing them.

### 6.5.1 Reduction in Vehicle Counts and Travel Distances

Figure 6.9 summarizes results of the primary objective of the CTSPAV for various vehicle capacities $K \in \{1, 2, 3, 4\}$. It reports the total number of vehicles needed to cover all the trips for each $K$ value by aggregating the final vehicle count results of every cluster. The number of vehicles utilized under *no-sharing* conditions (i.e., when the commuters travel using their personal vehicles) and under the original CTSP (with $K = 4$) (i.e., when drivers are selected from the set of commuters) are included for additional perspectives. The percentages in the figure report each count as a fraction of the no-sharing count. *The figure highlights the significant capability of the CTSPAV in reducing the number of vehicles used. Indeed, the CTSPAV reduces the vehicle counts by up to 92% every day*, and improves upon the original CTSP by an additional 34%. In fact, the results show that, even without any ride sharing (i.e., when $K = 1$), AVs still reduce the number of vehicles by 82% and improve upon the CTSP by an additional 24%. This reduction in vehicle counts can be translated into a significant reduction in parking spaces, which can then be utilized for other, more useful infrastructures. The difference in the vehicle counts of the CTSP and the CTSPAV is due to autonomy: The vehicles are not associated with drivers and can travel back and forth between the residential neighborhoods and the workplace to serve trips throughout the day. In the CTSP, the vehicles only perform a single inbound and outbound route every day as they are restricted to begin and end at the trip origins and destinations of their drivers.

Figure 6.10 summarizes the total travel distance of the vehicles, which is the secondary objective of the CTSPAV, under the same configurations. The results are again obtained by aggregating the results from every cluster, and the percentages represent each quantity as a fraction of the no-sharing total. The first result that stands out is how many more

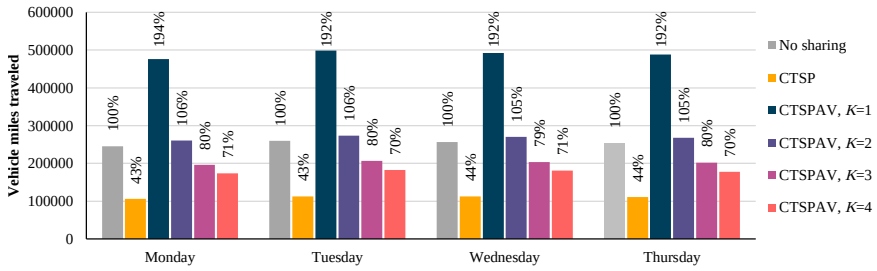



Figure 6.10: Total Travel Distance on Week 2.

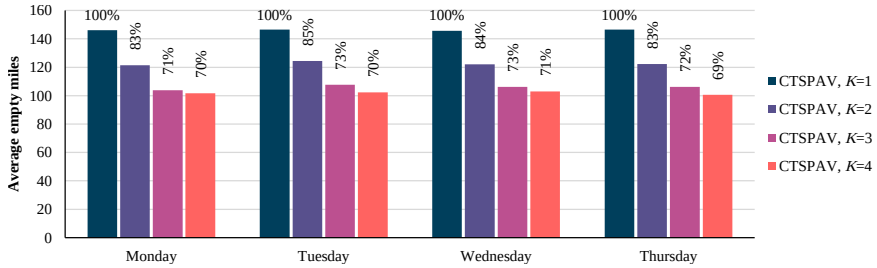



Figure 6.11: Average Empty Miles Per Vehicle on Week 2.

miles are traveled by the CTSPAV when $K = 1$ (92–94% more than those under the no-sharing conditions). When $K = 1$ for the CTSPAV, the AVs need to perform significantly more back-and-forth traveling between the neighborhoods and the workplace to cover the same number of trips, which consequently leads to their inflated total travel distance. The results improve significantly when $K$ is increased to 2 as the vehicles allow for more trip aggregation, yet the traveled miles are still 5–6% more than those of the private vehicles. Net savings in travel distance are only realized when $K \geq 3$: Beyond this point, *the reduction in travel distance from ride sharing exceeds the additional empty miles (the miles traveled by an AV with no passengers onboard) introduced by the back-and-forth traveling of the AVs.* Nevertheless, the 29–30% reduction in miles traveled when $K = 4$ is still not as significant as that offered by the original CTSP which is around 56–57%. Indeed, the CTSP does not introduce any empty miles and benefits from all the distance savings from ride sharing. On the other hand, the CTSPAV total will necessarily include some empty miles from when the vehicles travel without any passengers onboard as they go from the workplace back to the residential neighborhoods in the morning (or vice versa in the evening) to pick up more trips. There is obviously a trade-off between the reduction in vehicle count and travel distance. Figure 6.11 provides a closer look at the average empty miles per vehicle for the various vehicle capacities. The results are quite intuitive: The average decreases as $K$ increases, since the larger vehicle capacities allow for more ride sharing and require less back-and-forth traveling to cover the same number of trips.

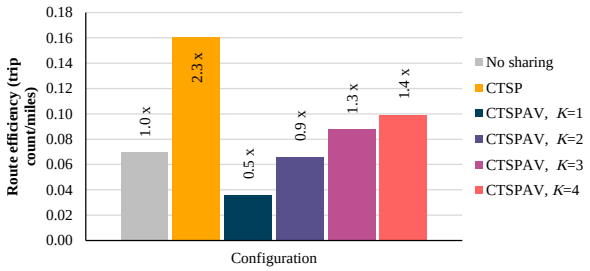
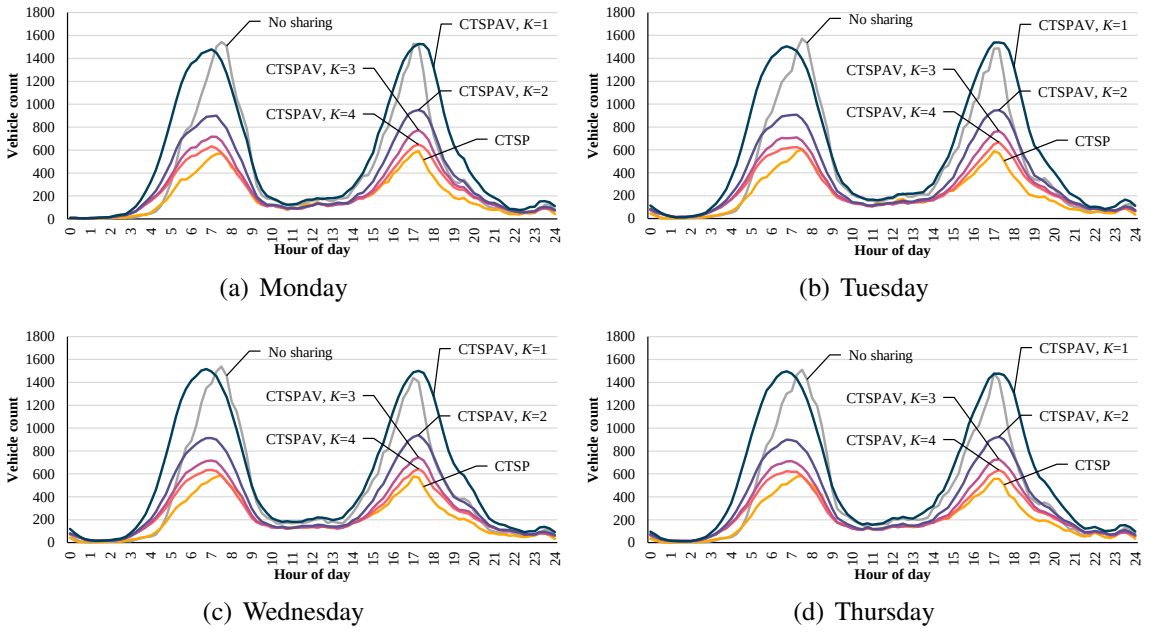



Figure 6.12: Efficiency of Vehicle Routes

Figure 6.12 then attempts to quantify the route efficiency of the various configurations, i.e., the number of trips covered per mile traveled. It also displays a multiplicative factor for each quantity as a multiple of the no-sharing value. The results indicate that the CTSP produces the most efficient routes, whereas the CTSPAV, when $K = 1$, is the least efficient. The CTSPAV gains more efficiency (albeit at a decreasing rate) as its vehicle capacity is increased; while its routes are more efficient than those of the private vehicles when $K = 4$, it still cannot outperform the CTSP. There is an intuitive explanation for this observation. The CTSPAV loses its route efficiency from its empty miles and then has to recover them by maximizing ride sharing to cover as many trips as possible. In contrast, the CTSP does not have to contend with any efficiency losses from empty miles.

### 6.5.2 Congestion Analysis

Figure 6.13 presents results on congestion to understand the reduction (or increase) in traffic caused by AVs compared to conventional vehicles. It tallies the total number of vehicles used by each configuration over every 15-minute interval throughout the four days considered. The goal is to investigate, qualitatively and comparatively, the capability of each configuration in flattening the traffic curve originally produced by the private vehicles. The CTSPAV with $K = 1$ appears to aggravate traffic as its curve is as tall as, and is wider than, that of the private vehicles. This is not surprising. As illustrated earlier, this configuration produces the most vehicle miles traveled and also the most empty miles. The curve is drastically flattened as soon as $K$ is increased to 2, and it keeps becoming flatter (at a decreasing rate) as $K$ is further increased. When $K = 4$, the CTSPAV produces about a 58% reduction in traffic at peak hours. The traffic curves of the CTSP appear to dominate slightly those of the CTSPAV with $K = 4$ most of the time. This observation is also in line with the route efficiency calculations. However, regardless of their relative performance, the figure provides the evidence that *both the CTSP and the CTSPAV have the potential to significantly reduce traffic congestion, especially during peak hours.*




(a) Monday (b) Tuesday

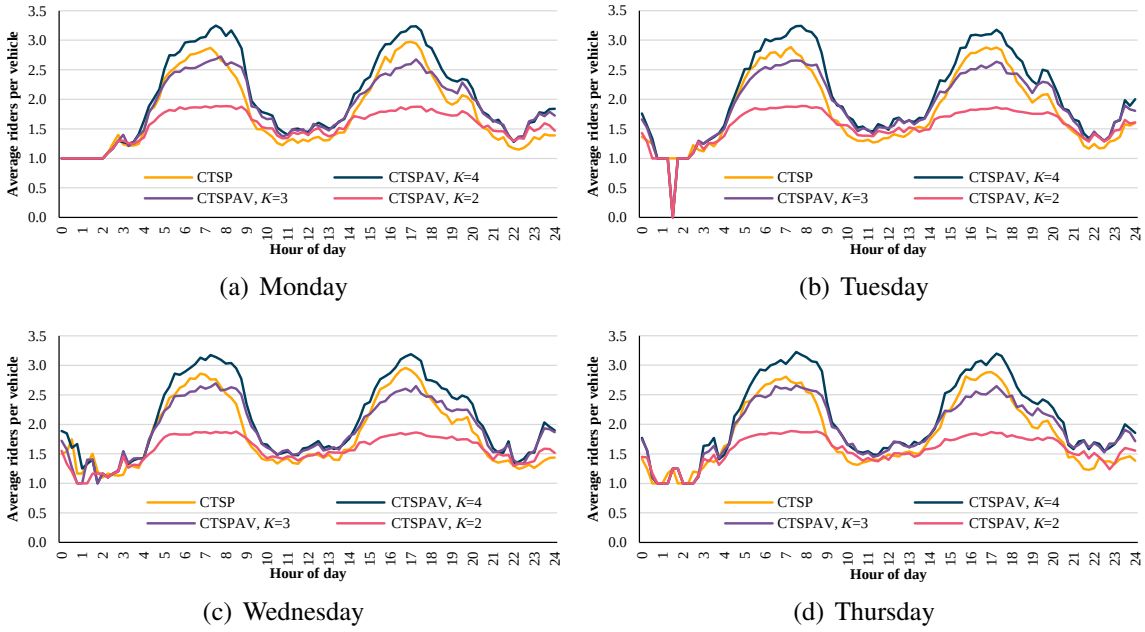
(c) Wednesday (d) Thursday

Figure 6.13: Number of Vehicles on the Road Over 15-Minute Intervals on Week 2.

### 6.5.3 Analysis of Commuting Properties

Figure 6.14 aims to quantify the relative amount of ride sharing taking place throughout each day for the different configurations. It reports the average number of riders per vehicle for every 15-minute interval throughout the four days considered. Results for the private vehicles and for the CTSPAV with $K = 1$ are not included for obvious reasons (they do not allow any sharing). The amount of ride sharing throughout a typical weekday mimics the shape of the trip demand: They both peak during the same periods of the day. This is to be expected as the CTSP and the CTSPAV maximize ride sharing, which is easier when the trip demand is higher. The figure also shows that the relative amount of sharing for the CTSPAV increases with the vehicle capacity. *Moreover, when $K = 4$, there is more ride sharing in the CTSPAV than in the CTSP most of the time.* This can be attributed to the higher flexibility of the mini routes of the CTSPAV compared to those of the CTSP. Indeed, a CTSP route must start and end at the origin and the destination of its driver's trip respectively, which constrains its total duration by the ride-duration limit of the driver. Mini routes of the CTSPAV are not subjected to these restrictions, allowing for more flexibility in serving trips. Interestingly, during peak hours, the average number of riders per vehicle is between 3.0 and 3.5 (for the CTSPAV with $K = 4$) due to the spatial and temporal characteristics of the trips from the dataset. This also indicates the types of AVs that will be most useful in the future, at least for cities like Ann Arbor, do not require large capacities.

Figure 6.15 reports the average commute times, i.e., the average time spent on the




(a) Monday (b) Tuesday

(c) Wednesday (d) Thursday

Figure 6.14: Average Riders Per Vehicle Over 15-Minute Intervals on Week 2

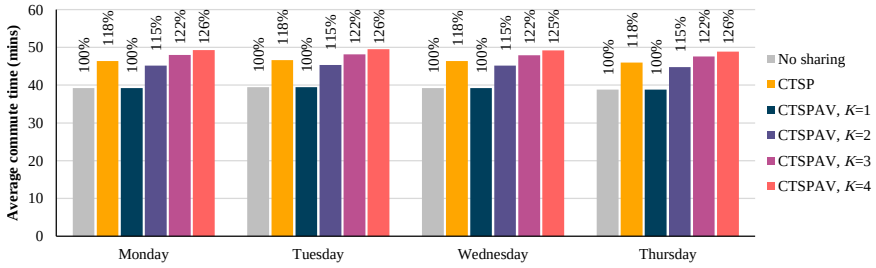



Figure 6.15: Average Commute Time on Week 2.

vehicle by each rider. The percentages of each quantity are calculated relative to the no-sharing value. The results shed light on another inherent trade-off in ride-sharing services as the ride duration necessarily increases. During ride sharing, a route may deviate from the optimal path to pick up or drop off other riders. This, combined with possible wait times incurred at the pickup locations, contributes to the increased ride durations. The results reveal an expected trend for the CTSPAV: The average commute times increase with the increase in vehicle capacity. However, *it is interesting to observe that, although parameter $R$ was set to 50% for the case study, the commute times of the CTSPAV with $K = 4$ only increase by an average of 26%. The CTSPAV thus guarantees a high QoS for its riders.*

## 6.6 Conclusion

The purpose of the CTSPAV is to synthesize an optimal routing plan for serving a large set of commute trips with AVs. Its design was originally motivated by the desire to address the growing parking and traffic congestion problems induced by an average of 9,000 daily commuters traveling to parking lots operated by the University of Michigan located in downtown Ann Arbor, Michigan. Utilization of AVs was seen as the key to addressing the shortcomings of the original CTSP—a conventional car-pooling problem with the same objectives as the CTSPAV—by obviating any driver-related requirements that could limit its ride-sharing potential. A first attempt at solving the problem in Chapter 5 investigated two different methods: (1) A CTSPAV procedure which used column-generation to discover mini routes—short routes covering only inbound or outbound trips that have distinct pickup, transit, and drop-off phases—with negative reduced costs which are chained together to form longer AV routes in its master problem and (2) A DARP procedure which uses a classical column-generation approach originally developed for the DARP to solve the CTSPAV. Both methods utilized identical lexicographic objectives which sought to first minimize the required vehicle count and then minimize their total travel distance. To deal with the complexity of handling the massive volume of trips, the commuters were first clustered into groups representing artificial neighborhoods, after which ride sharing within each cluster was optimized exclusively. We discovered that each method had a trade-off. The CTSPAV procedure produced strong integer solutions but had weak primal lower bounds. Conversely, the DARP procedure generated stonger primal lower bounds especially for the primary objective, but it was slow and therefore could not obtain strong integer solutions within time-constrained scenarios.

*The trade-offs of the two procedures presented an opportunity for exploring a method that could leverage the strengths of both, which is the primary methodological contribution of this work.* This work thus proposes a branch-and-cut procedure that exploits a dual-modeling approach for solving the CTSPAV. The core of the procedure is a MIP formulation of the CTSPAV that chains (exhaustively enumerated) mini routes to form longer AV routes and that is capable of producing high-quality integer solutions. This core is complemented by a DARP formulation (for minimizing vehicle counts) whose relaxation is obtained through a column-generation procedure. The DARP formulation is less effective at finding high-quality integer solutions, but its relaxation produces strong lower bounds. The overall algorithm solves the core branch-and-cut procedure and the DARP relaxation in parallel, transmitting new lower bounds asynchronously from the relaxation to the branch-and-cut procedure. Computational evaluations that use instances derived from

the Ann Arbor commute-trip dataset demonstrated that this hybrid algorithm consistently outperforms a similar branch-and-cut procedure that utilizes other well-established valid inequalities like 2-path cuts and successor and predecessor inequalities. It successfully closed the optimality gaps for several large- and medium-sized instances as well as those for all tight problem instances considered in the evaluation, of which none could be optimally solved by the CTSPAV procedure from Chapter 5.

With the availabilty of the exact branch-and-cut procedure, this work then provided a comprehensive analysis of the potential of the AV ride-sharing platform in relieving parking pressure and traffic congestion for a medium-sized city. In particular, the work presented the results of a case study which applies the clustering-CTSPAV optimization workflow on the large-scale dataset of commute trips from the city of Ann Arbor, Michigan. *The analysis revealed several invaluable insights, including the CTSPAV's capability of reducing daily vehicle counts by 92%, further improving upon the already massive 57% vehicle reductions of the original CTSP*. It does so by generating AV routes that are very long—a stark contrast to the short routes of the CTSP—allowing each AV to cover significantly more trips every day. It could also effectively flatten the vehicle usage curve (which represents the number of vehicles used per unit time), suggesting a concomitant ability to effectively reduce traffic congestion. The CTSPAV also produced higher averages of trips shared per unit time than the CTSP, indicating that it is superior at aggregating trips for ride sharing. The analysis also revealed some drawbacks, the most significant being the introduction of empty miles into the daily travel-distance totals. The empty miles degrade the efficiency of the CTSPAV routes, which measures the average number of trips covered per distance traveled, making them less efficient than the routes of the CTSP. Empty miles are unfortunately a by-product that is inherent to the utilization of AVs, and its introduction is a trade-off that will need to be carefully weighed against the benefits of AVs by the ride-sharing platform operator. Nonetheless, the case study indicated that the CTSPAV routing plan, even with its empty miles, is still able to reduce the total miles traveled by 30% while producing routes that are 1.4 times more efficient than those of private vehicles. On the whole, the case study showed that a CTSPAV-based ride-sharing platform could significantly reduce daily vehicle counts and the number of vehicles used per unit time. Such a platform would be highly effective at aggregating trips, making it a very promising solution for reducing parking space utilization and for mitigating traffic congestion induced by large-scale commuting.

# CHAPTER 7

# Conclusion

This work began from a collaboration between MIDAS and the LTP division of the University of Michigan that sought to relieve the parking pressure induced by the 15,000 commuters traveling to the 15 university-operated parking structures located in downtown Ann Arbor, Michigan. The desire to explore the potential of optimized car-pooling and ride-sharing platforms in reducing this pressure led to a series of studies that focused on tackling the computational complexities of numerous aspects of the optimization problem. These studies have culminated in this dissertation, which describes the development of mathematical models, decomposition approaches, and algorithmic techniques to solve numerous vehicle-routing problems that are tailored for commuting, ranging from selecting and scheduling drivers to pick up and drop off other riders on their way to work and back home, to designing optimal routes for fleets of autonomous vehicles to cover the commute trips. Every solution approach proposed in this work shares a common characteristic: Each relies on LP-based techniques that produce valid lower bounds to its objective value (unlike typical heuristics), which lets the solution quality be quantified through the calculation of an optimality gap. They are also bolstered by the availability of the Ann Arbor commute-trip dataset, which allowed the algorithms to be evaluated on real-world data, and whose analyzed results provided invaluable insight into the performance characteristics of the optimized routing plans.

The dissertation begins with three chapters that consider the optimization of routing plans whereby the trips are served by conventional vehicles driven by the commuters themselves. Chapter 2 first explores several optimization models that adopt different driver- and passenger-matching requirements aimed at satisfying different guiding principles that are deemed desirable for a ride-sharing platform. It then considers a clustering-route enumeration-MIP solving workflow to solve each model. Its computational results revealed that the model demanding the highest amount of flexibility in role adoption and sharing preferences also permitted the most trip sharing and hence produced the best ve-

hicle reduction potential. Chapter 3 then selects and expands upon the best performing model, formalizing it as the CTSP. It also proposes an exact and an approximate algorithm for solving the model, both utilizing column generation, to scale the solutions to larger vehicle capacities and to more riders. An analysis of its computational results showed the model's ability to reduce the commuting vehicle count in Ann Arbor by up to 57%, and it also revealed that the inherently short nature of the model's routes was limiting its ability to further aggregate trips. Chapter 4 approaches the CTSP from a realistic, operational perspective in which commuters may decide to alter their return trip schedules due to unforeseen circumstances. It proposes a two-stage optimization approach that incorporates a scenario-sampling technique to select the set of drivers that can best respond to this schedule uncertainty. A simulation using real-world data demonstrated the approach's ability to increase plan robustness by utilizing more sampled scenarios, and it also revealed a robustness-vehicle reduction trade-off that can be optimally evaluated by comparing the per-unit prices of vehicle increase and uncovered riders.

The next two chapters approach the CTSP from a different angle. Driven by the desire to address the key factor—the driver induced constraints—that was limiting further reduction of the vehicle counts, and to capitalize on the emergence of autonomous-driving technology, the chapters consider the problem of optimizing fleets of autonomous vehicles to serve the trips of the CTSP. Chapter 5 formalizes this problem as the CTSPAV and contrasts two approaches to tackle the complexity of designing the anticipated longer AV routes. The first attempts to leverage the spatial structure of the trip data by using a CTSPAV MIP formulation that chains together short, single-direction mini routes, while the second uses a more traditional DARP formulation. Each was then solved with a formulation-specific column-generation procedure, and subsequent computational comparisons revealed that each had complementary performance trade-offs: The CTSPAV procedure finds strong integer solutions quickly but has weak lower bounds, whereas the DARP procedure produces strong primal and dual lower bounds, but it is slower and thus develops weaker integer solutions. This discovery led to the work in Chapter 6 which aims to leverage the unique strengths of both approaches by melding them together. It resulted in a branch-and-cut procedure that solves the CTSPAV MIP and retrieves vehicle-count lower bounds from a column-generation procedure that solves a DARP formulation of the same problem in parallel. The computational evaluations demonstrated the efficacy of this novel dual-modeling approach, whereby it outperformed another branch-and-cut procedure that relied on other families of valid inequalities and the CTSPAV column-generation procedure from Chapter 5. Further analysis of its routing plans revealed that the CTSPAV is not only able to reduce commuting vehicle counts in Ann Arbor by 92%, but it also significantly reduces (by 58%) the number

of vehicles on the road during peak hours, which bodes very well for its ability to ease traffic congestion. An inherent drawback to using AVs, however, is that they introduce some inefficiencies in terms of empty miles when they sometimes travel without any passengers onboard.

On the whole, the case studies have demonstrated that there are numerous potential benefits that can be unlocked through the use of optimized ride-sharing platforms for commuting, ranging from reductions to the daily number of vehicles required—and hence to the total demand for parking spaces—to decreases to the number of vehicles on the road during peak hours—and hence to the traffic congestion over the same period. When using conventional vehicles, our study showed that a platform that optimizes routing plans on a daily basis—by selecting different sets of drivers and having passengers share rides with different people on their trips to the workplace and back home every day—is needed to permit the maximum amount of trip aggregation for the varying daily schedules. Such a platform was shown to be capable of reducing daily vehicle counts and vehicle miles traveled for the commute trips from Ann Arbor by approximately 57% while also decreasing peak vehicle usage by 62%. When using AVs that depart from and return to a centralized depot, the daily vehicle counts can be further reduced by 92% while decreasing peak vehicle usage by 58%. Reductions to vehicle miles traveled are more modest at only 30% as the AVs introduce some empty miles into their travel distance totals. These benefits, however, are not free. They come at the price of numerous psychological discomforts and inconveniences that have to be tolerated by the riders. They include increased commute times, having to potentially travel with different people on every trip, reduced flexibility in trip choices, and having to plan and book trips in advance. There also exist trade-offs between the benefits of conventional and autonomous vehicles for the platform: While the vehicle reduction capability of the AVs are unmatched, their routes are also significantly less efficient than those of the conventional vehicles. The AV routes cover longer distances per trip served as their distance totals will necessarily include some empty miles that are introduced when the vehicles travel without any passengers onboard.

From a problem-solving/optimization perspective, a careful analysis of the problem constraints and of the characteristics (e.g. spatial and temporal) of the input data to unearth structures/patterns inherent to the problem that can be exploited by a solution approach remains a pivotal factor to effectively solve NP-hard VRPs such as those considered in this work. For instance, the key to effectively solve the CTSP was to recognize that its routes will travel in a single direction (inbound/outbound) and that each will be bookended by the origin and the destination of its driver, and how this constraint would, in turn, limit the length of the routes. This recognition drove the synthesis of the various proposed solution

approaches: It not only affected the formulation of the CTSP MIP, but it also led to the conception of the REA which could exhaustively enumerate the routes in polynomial time, and it influenced the design of the PSPs of the column-generation approaches, whereby they consider each commuter as a driver and searched for the optimal routes driven by each by solving multiple independent ESPPRCs. On the other hand, the key to solving the CTSPAV was to recognize the existence of the hub-and-spoke-like spatial structure in the origins and the destinations of the trips, and to realize that the structure is common not only to the commute trips in most American cities, but also to other problems like the last-mile transportation problem which involves shuttling passengers to and from the transportation hubs of a multi-modal system (Raghunathan et al. 2018b,a, Mahéo et al. 2019, Pinto et al. 2020, Basciftci and Van Hentenryck 2020). This structure innately decomposes a long AV route into shorter constituents that each travel in a single direction (inbound/outbound) and that each visits only origins followed by only destinations. This decomposition allowed the derivation of the CTSPAV MIP that exploits the structure by using a formulation that chains together feasible mini routes to form the longer AV routes, the conception of the MREA that exhaustively enumerates the mini routes in polynomial time, and the design of the PSP of the column-generation procedure which only searches for optimal mini routes. In addition to exploiting inherent problem structures, any effective solution approach to large-scale optimization problems will also necessitate the evaluation of different trade-offs to the available solution techniques, several of which have been explored in this work. Examples of some trade-offs that have been considered include the trade-off between computational speed and proving optimality, whereby a solution approach that is required to complete quickly is typically unable to close its optimality gap within its allocated time limit, and the trade-off between the effectiveness of a decomposition approach and its generalizability, whereby a decomposition approach that is tailored to specifically exploit a problem structure makes it more effective at solving the problem, but it also makes the approach less applicable to more general problems that do not exhibit the structure.

Even though tremendous strides have been made in this dissertation to design effective solution approaches for the CTSP and the CTSPAV, there remains a vast room for improvement that presents opportunities for future research. Some potential future research directions include:

1. *Role of heuristics*: Heuristics will continue to play an important role in solving VRPs, especially in operational settings where computational speed is of the paramount importance compared to proving optimality or quantifying solution quality, e.g., when dealing with dynamic requests where the problem needs to be solved quickly at a regular frequency. Studying their potential in solving the various subproblems of the

proposed solution approaches and contrasting their trade-offs against the exact methods used in this work would therefore be very illuminating, e.g.: (1) Using the Lin-Kernighan heuristic (Lin and Kernighan 1973, Helsgaun 2000) to discover negative reduced cost routes from the PSPs of the numerous column-generation procedures, (2) Using the Wedelin heuristic (Wedelin 1995, Bastert et al. 2010) to solve the numerous MIPs to find feasible integer solutions, or (3) Using tabu search (Cordeau and Laporte 2003b) to solve the CTSP or the CTSPAV in its entirety.

2. *Applicability to metropolitan areas*: The experimental evaluations in this study have thus far focused on the trips from a medium-sized city. It would therefore be interesting to not only perform a similar case study on metropolitan areas, but to also investigate the applicability of some of the solution approaches that was tailored specifically for the Ann Arbor dataset, e.g., the CTSPAV procedures that leveraged the hub-and-spoke spatial structure of the trips. The absence of such a structure would provide opportunities to explore other creative ways of grouping trips so that they would benefit the CTSPAV procedures, or to analyze the trip data to unearth other exploitable structures and to tailor new decomposition approaches for them.
3. *Schedule uncertainty for the CTSPAV*: While a scenario-sampling approach was proposed to handle such uncertainties for the CTSP, a similar study have yet to be performed on the CTSPAV. The absence of drivers and the longer nature of the routes in the CTSPAV will necessitate a different perspective and approach to handling the issue. Such a study will also require the development of a real-time simulator that keeps track of the state of the entire ride-sharing system at every moment, e.g., the position of every vehicle, the trips being served by them, and the set of trips that have and have not been completed throughout the day.
4. *Social acceptance studies*: While this dissertation has focused on the algorithmic side to operating a ride-sharing service, there is another side to the problem that is just as equally important: the social aspect that needs to be well understood before such a service can be successfully deployed. Studying this side will require numerous surveys combined with pilot and user-acceptance tests to better understand user preferences for various aspects including, but not limited to, their reception to the idea of potentially traveling with strangers every day, convenient mechanisms for booking trips, and acceptable time-window and ride-duration limit tolerances.

# APPENDIX A

# Appendix for Chapter 3

## A.1 Computational Results

Results of the vehicle capacity scaling experiments for the REA are presented in Table A.1. The first three columns show the cluster size, vehicle capacity, and cluster ID which characterize each problem instance. The next column lists the number of columns generated by the algorithm, while the following three show the results of solving the MP with a MIP solver. They show the vehicle count, the total distance of selected routes, and the optimality gap of the MIP solution. Finally, the remaining two columns display computation times of the route-enumeration phase and of the entire algorithm including the MIP solve times.

Table A.1: Results of REA Scalability with Increasing Vehicle Capacity ($\Delta = 10$ mins, $R = 50\%$).

| Cluster size | Vehicle capacity | Cluster ID | Column # | Vehicle # | Total distance (m) | Optimality gap (%) | Wall time (s) | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | Route enumeration | Total |
| 75 | 4 | C0-75 | 508 | 47 | 356296 | 0.00 | 11 | 11 |
| | | C2-75 | 1946 | 40 | 643541 | 0.00 | 9 | 9 |
| | | C3-75 | 2068 | 38 | 487061 | 0.00 | 11 | 11 |
| | | C4-75 | 1483 | 43 | 398265 | 0.00 | 10 | 10 |
| | | C5-75 | 274 | 61 | 230460 | 0.00 | 9 | 9 |
| | | C6-75 | 3071 | 36 | 451262 | 0.00 | 11 | 12 |
| | | C7-75 | 690 | 46 | 437743 | 0.00 | 10 | 10 |
| | | C8-75 | 1350 | 37 | 487561 | 0.00 | 10 | 10 |
| | | C9-75 | 3926 | 31 | 328525 | 0.00 | 10 | 37 |
| | | C10-75 | 4137 | 32 | 527544 | 0.00 | 13 | 14 |
| | | C11-75 | 866 | 47 | 443770 | 0.00 | 12 | 12 |
| | | C12-75 | 592 | 46 | 298162 | 0.00 | 11 | 11 |
| | | C13-75 | 2143 | 37 | 481869 | 0.00 | 12 | 12 |
| | | C14-75 | 863 | 47 | 301108 | 0.00 | 10 | 10 |
| | | C15-75 | 1512 | 36 | 460087 | 0.00 | 10 | 10 |
| | | C17-75 | 457 | 50 | 348581 | 0.00 | 9 | 9 |
| | | C19-75 | 496 | 57 | 245827 | 0.00 | 10 | 10 |
| | | C20-75 | 752 | 46 | 420825 | 0.00 | 10 | 10 |
| | | C22-75 | 2288 | 40 | 472247 | 0.00 | 10 | 11 |
| | | C23-75 | 1574 | 36 | 385434 | 0.00 | 9 | 10 |
| | | C24-75 | 1926 | 37 | 425807 | 0.00 | 11 | 12 |
| | | C26-75 | 468 | 52 | 333692 | 0.00 | 9 | 9 |
| | | C28-75 | 750 | 44 | 293838 | 0.00 | 10 | 10 |
| | | C29-75 | 2541 | 32 | 548779 | 0.00 | 10 | 11 |
| | 5 | C0-75 | 509 | 47 | 356296 | 0.00 | 306 | 306 |
| | | C2-75 | 2014 | 40 | 643540 | 0.00 | 316 | 316 |
| | | C3-75 | 2117 | 38 | 487061 | 0.00 | 341 | 342 |
| | | C4-75 | 1559 | 43 | 398216 | 0.00 | 326 | 326 |
| | | C5-75 | 274 | 61 | 230460 | 0.00 | 264 | 264 |
| | | C6-75 | 3331 | 35 | 450414 | 0.00 | 355 | 356 |
| | | C7-75 | 690 | 46 | 437743 | 0.00 | 316 | 316 |
| | | C8-75 | 1352 | 37 | 487245 | 0.00 | 335 | 335 |
| | | C9-75 | 4110 | 31 | 327993 | 0.00 | 346 | 378 |
| | | C10-75 | 5589 | 32 | 526188 | 0.00 | 352 | 354 |
| | | C11-75 | 870 | 47 | 443770 | 0.00 | 300 | 300 |
| | | C12-75 | 592 | 46 | 298162 | 0.00 | 308 | 308 |
| | | C13-75 | 2207 | 37 | 479166 | 0.00 | 351 | 351 |
| | | C14-75 | 867 | 47 | 301108 | 0.00 | 304 | 304 |
| | | C15-75 | 1522 | 36 | 460087 | 0.00 | 335 | 335 |
| | | C17-75 | 457 | 50 | 348581 | 0.00 | 293 | 293 |
| | | C19-75 | 496 | 57 | 245827 | 0.00 | 279 | 279 |
| | | C20-75 | 752 | 46 | 420825 | 0.00 | 309 | 309 |
| | | C22-75 | 2344 | 40 | 472040 | 0.00 | 347 | 348 |
| | | C23-75 | 1628 | 35 | 379234 | 0.00 | 306 | 307 |
| | | C24-75 | 1996 | 37 | 425806 | 0.00 | 339 | 339 |
| | | C26-75 | 468 | 52 | 333692 | 0.00 | 292 | 292 |
| | | C28-75 | 751 | 44 | 293838 | 0.00 | 315 | 315 |
| | | C29-75 | 2594 | 32 | 543850 | 0.00 | 350 | 351 |
| | 6 | C0-75 | 509 | 47 | 356296 | 0.00 | 4761 | 4761 |
| | | C2-75 | 2016 | 40 | 643540 | 0.00 | 5109 | 5110 |
| | | C3-75 | 2120 | 38 | 487061 | 0.00 | 6009 | 6009 |
| | | C4-75 | 1562 | 43 | 398216 | 0.00 | 5375 | 5375 |
| | | C5-75 | 274 | 61 | 230460 | 0.00 | 3220 | 3220 |
| | | C6-75 | 3406 | 35 | 450414 | 0.00 | 6341 | 6341 |
| | | C7-75 | 690 | 46 | 437743 | 0.00 | 6135 | 6135 |
| | | C8-75 | 1352 | 37 | 487245 | 0.00 | 7172 | 7172 |
| | | C9-75 | 4116 | 31 | 327993 | 0.00 | 7893 | 7916 |
| | | C10-75 | 6571 | 32 | 525115 | 0.00 | 7480 | 7482 |
| | | C11-75 | 870 | 47 | 443770 | 0.00 | 5413 | 5413 |
| | | C12-75 | 592 | 46 | 298162 | 0.00 | 5206 | 5206 |
| | | C13-75 | 2209 | 37 | 479166 | 0.00 | 6164 | 6164 |
| | | C14-75 | 867 | 47 | 301108 | 0.00 | 4658 | 4658 |
| | | C15-75 | 1522 | 36 | 460087 | 0.00 | 5668 | 5668 |
| | | C17-75 | 457 | 50 | 348581 | 0.00 | 4147 | 4147 |
| | | C19-75 | 496 | 57 | 245827 | 0.00 | 4000 | 4000 |
| | | C20-75 | 752 | 46 | 420825 | 0.00 | 4983 | 4983 |
| | | C22-75 | 2345 | 40 | 472040 | 0.00 | 5961 | 5961 |
| | | C23-75 | 1629 | 35 | 379234 | 0.00 | 5008 | 5008 |
| | | C24-75 | 2001 | 37 | 425806 | 0.00 | 5843 | 5844 |
| | | C26-75 | 468 | 52 | 333692 | 0.00 | 4254 | 4254 |
| | | C28-75 | 751 | 44 | 293838 | 0.00 | 4935 | 4935 |
| | | C29-75 | 2594 | 32 | 543850 | 0.00 | 6071 | 6071 |

| Cluster size | Vehicle capacity | Cluster ID | Column # | Vehicle # | Total distance (m) | Optimality gap (%) | Wall time (s) | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | Route enumeration | Total |
| 100 | 4 | C0-100 | 854 | 63 | 488119 | 0.00 | 34 | 34 |
| | | C1-100 | 456 | 75 | 331497 | 0.00 | 31 | 31 |
| | | C2-100 | 3802 | 46 | 596824 | 0.00 | 36 | 36 |
| | | C3-100 | 4510 | 46 | 586434 | 0.00 | 35 | 36 |
| | | C4-100 | 4232 | 44 | 558323 | 0.00 | 35 | 41 |
| | | C5-100 | 732 | 70 | 366565 | 0.00 | 31 | 31 |
| | | C6-100 | 2579 | 47 | 724844 | 0.00 | 35 | 35 |
| | | C7-100 | 2020 | 53 | 661648 | 0.00 | 34 | 34 |
| | | C8-100 | 1803 | 51 | 609120 | 0.00 | 34 | 34 |
| | | C9-100 | 12034 | 40 | 527217 | 0.00 | 36 | 38 |
| | | C10-100 | 1095 | 58 | 426386 | 0.00 | 34 | 34 |
| | | C11-100 | 2374 | 51 | 427829 | 0.00 | 34 | 35 |
| | | C12-100 | 989 | 62 | 413012 | 0.00 | 33 | 33 |
| | | C13-100 | 909 | 61 | 483087 | 0.00 | 34 | 34 |
| | | C14-100 | 4306 | 40 | 693825 | 0.00 | 35 | 40 |
| | | C15-100 | 4605 | 48 | 790005 | 0.00 | 34 | 37 |
| | | C16-100 | 1578 | 55 | 627423 | 0.00 | 33 | 33 |
| | | C17-100 | 1151 | 60 | 640945 | 0.00 | 33 | 33 |
| | | C18-100 | 952 | 58 | 681240 | 0.00 | 34 | 34 |
| | | C19-100 | 2226 | 51 | 503252 | 0.00 | 34 | 35 |
| | | C20-100 | 4254 | 46 | 569724 | 0.00 | 35 | 36 |
| | | C21-100 | 667 | 78 | 328216 | 0.00 | 32 | 32 |
| | 5 | C0-100 | 854 | 63 | 488119 | 0.00 | 1185 | 1186 |
| | | C1-100 | 456 | 75 | 331497 | 0.00 | 1072 | 1072 |
| | | C2-100 | 3910 | 46 | 596658 | 0.00 | 1356 | 1357 |
| | | C3-100 | 4814 | 46 | 584077 | 0.00 | 1357 | 1359 |
| | | C4-100 | 4487 | 44 | 552862 | 0.00 | 1298 | 1342 |
| | | C5-100 | 735 | 70 | 366565 | 0.00 | 1034 | 1034 |
| | | C6-100 | 2596 | 47 | 724844 | 0.00 | 1362 | 1362 |
| | | C7-100 | 2043 | 53 | 661528 | 0.00 | 1326 | 1326 |
| | | C8-100 | 1842 | 51 | 609038 | 0.00 | 1317 | 1317 |
| | | C9-100 | 16749 | 40 | 519564 | 0.00 | 1419 | 1451 |
| | | C10-100 | 1096 | 58 | 426386 | 0.00 | 1254 | 1254 |
| | | C11-100 | 2418 | 51 | 427829 | 0.00 | 1320 | 1321 |
| | | C12-100 | 990 | 62 | 413012 | 0.00 | 1208 | 1208 |
| | | C13-100 | 912 | 61 | 483087 | 0.00 | 1221 | 1221 |
| | | C14-100 | 4448 | 40 | 689319 | 0.00 | 1383 | 1457 |
| | | C15-100 | 5985 | 48 | 788340 | 0.00 | 1287 | 1294 |
| | | C16-100 | 1599 | 55 | 626787 | 0.00 | 1226 | 1227 |
| | | C17-100 | 1155 | 60 | 640945 | 0.00 | 1218 | 1218 |
| | | C18-100 | 953 | 58 | 681240 | 0.00 | 1292 | 1293 |
| | | C19-100 | 2291 | 51 | 503252 | 0.00 | 1352 | 1352 |
| | | C20-100 | 4761 | 46 | 568797 | 0.00 | 1352 | 1354 |
| | | C21-100 | 667 | 78 | 328216 | 0.00 | 1080 | 1080 |
| | 6 | C0-100 | 854 | 63 | 488119 | 0.00 | 25088 | 25088 |
| | | C1-100 | 456 | 75 | 331497 | 0.00 | 20117 | 20117 |
| | | C2-100 | 3917 | 46 | 596658 | 0.00 | 33455 | 33456 |
| | | C3-100 | 4872 | 46 | 584048 | 0.00 | 33801 | 33802 |
| | | C4-100 | 4510 | 44 | 552862 | 0.00 | 29993 | 30180 |
| | | C5-100 | 735 | 70 | 366565 | 0.00 | 18613 | 18613 |
| | | C6-100 | 2598 | 47 | 724844 | 0.00 | 43135 | 43136 |
| | | C7-100 | 2043 | 53 | 661528 | 0.00 | 38791 | 38792 |
| | | C8-100 | 1848 | 51 | 609038 | 0.00 | 39437 | 39437 |
| | | C9-100 | 18758 | 40 | 517624 | 0.00 | 48359 | 48403 |
| | | C10-100 | 1096 | 58 | 426386 | 0.00 | 36941 | 36941 |
| | | C11-100 | 2419 | 51 | 427829 | 0.00 | 39761 | 39763 |
| | | C12-100 | 990 | 62 | 413012 | 0.00 | 31038 | 31038 |
| | | C13-100 | 912 | 61 | 483087 | 0.00 | 33881 | 33881 |
| | | C14-100 | 4456 | 40 | 689319 | 0.00 | 41455 | 41492 |
| | | C15-100 | 6939 | 48 | 788340 | 0.00 | 36980 | 36990 |
| | | C16-100 | 1600 | 55 | 626787 | 0.00 | 32031 | 32031 |
| | | C17-100 | 1155 | 60 | 640945 | 0.00 | 34810 | 34810 |
| | | C18-100 | 953 | 58 | 681240 | 0.00 | 38501 | 38501 |
| | | C19-100 | 2294 | 51 | 503252 | 0.00 | 42573 | 42574 |
| | | C20-100 | 4898 | 46 | 568797 | 0.00 | 43900 | 43903 |
| | | C21-100 | 667 | 78 | 328216 | 0.00 | 29507 | 29507 |

Table A.2 shows results of the same set of experiments for the BPA. Similar to Table A.1, the first three columns list the cluster size, vehicle capacity, and cluster ID for each problem instance. The next two columns present the total number of unique feasible edges from all inbound and outbound graphs to further characterize the size of the problem instances, while the following two show the total number of tree nodes explored and columns generated by the algorithm. The next two display the results in terms of the vehicle count and the total distance of selected routes. The following two columns list optimality gaps of the MIP solution at the root node and of the best feasible solution, while the next lists the integrality gap of the best feasible solution, which is the relative gap between its objective value and $z^*$. Finally, the remaining four columns show computation times for the RMP to converge, for finding the MIP solution at the root node, for arriving at the best feasible solution, and for executing the entire algorithm.

Table A.2: Results of BPA Scalability with Increasing Vehicle Capacity ($\Delta = 10$ mins, $R = 50\%$).

| Cluster size | Vehicle capacity | Cluster ID | Inbound edge # | Outbound edge # | Tree node # | Column # | Vehicle # | Total distance (m) | Optimality gap (%) | | Integrality gap (%) | Wall time (s) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | Root node soln. | Best feasible soln. | | RMP conv. | Root node soln. | Best feasible soln. | Total |
| 75 | 4 | C0-75 | 1661 | 1565 | 15 | 462 | 47 | 356296 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C2-75 | 1906 | 1405 | 57 | 1043 | 40 | 643541 | 0.01 | 0.00 | 0.01 | 2 | 2 | 7 | 12 |
| | | C3-75 | 3099 | 1809 | 31 | 1247 | 38 | 487061 | 0.00 | 0.00 | 0.00 | 2 | 2 | 12 | 13 |
| | | C4-75 | 2052 | 1851 | 79 | 915 | 43 | 398265 | 0.01 | 0.00 | 0.01 | 1 | 1 | 8 | 12 |
| | | C5-75 | 823 | 810 | 1 | 267 | 61 | 230460 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C6-75 | 3427 | 2012 | 17 | 1420 | 36 | 451262 | 2.77 | 0.00 | 2.77 | 4 | 5 | 5 | 16 |
| | | C7-75 | 2458 | 1649 | 25 | 615 | 46 | 437743 | 2.17 | 0.00 | 2.17 | 1 | 1 | 1 | 2 |
| | | C8-75 | 3002 | 2183 | 27 | 1028 | 37 | 487561 | 2.69 | 0.00 | 2.69 | 2 | 2 | 11 | 15 |
| | | C9-75 | 3550 | 2341 | 428 | 2613 | 31 | 328525 | 3.22 | 0.00 | 3.21 | 7 | 8 | 493 | 528 |
| | | C10-75 | 2856 | 1580 | 151 | 2085 | 32 | 527544 | 3.11 | 0.00 | 3.11 | 11 | 11 | 168 | 232 |
| | | C11-75 | 1955 | 1193 | 23 | 639 | 47 | 443770 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 2 |
| | | C12-75 | 1801 | 1347 | 5 | 501 | 46 | 298162 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C13-75 | 2699 | 1302 | 141 | 1392 | 37 | 481869 | 2.70 | 0.00 | 2.70 | 2 | 2 | 17 | 37 |
| | | C14-75 | 2045 | 1332 | 17 | 626 | 47 | 301108 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 2 |
| | | C15-75 | 3007 | 1443 | 23 | 1098 | 36 | 460087 | 0.01 | 0.00 | 0.01 | 2 | 2 | 2 | 11 |
| | | C17-75 | 1492 | 835 | 3 | 404 | 50 | 348581 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C19-75 | 1497 | 664 | 1 | 390 | 57 | 245827 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C20-75 | 2323 | 1254 | 17 | 683 | 46 | 420825 | 0.00 | 0.00 | 0.00 | 1 | 1 | 2 | 2 |
| | | C22-75 | 3113 | 1524 | 119 | 1373 | 40 | 472247 | 2.49 | 0.00 | 2.49 | 2 | 2 | 17 | 45 |
| | | C23-75 | 2069 | 1548 | 169 | 1105 | 36 | 385434 | 5.53 | 0.00 | 5.53 | 2 | 2 | 28 | 45 |
| | | C24-75 | 2446 | 1694 | 305 | 1479 | 37 | 425807 | 2.70 | 0.00 | 2.69 | 2 | 3 | 72 | 94 |
| | | C26-75 | 1957 | 1039 | 17 | 432 | 52 | 333692 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C28-75 | 1781 | 1452 | 5 | 611 | 44 | 293838 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C29-75 | 2495 | 2238 | 1242 | 2217 | 32 | 548779 | 3.03 | 0.00 | 0.02 | 4 | 19 | 623 | 805 |
| | 5 | C0-75 | 1661 | 1565 | 15 | 462 | 47 | 356296 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C2-75 | 1906 | 1405 | 115 | 1085 | 40 | 643540 | 0.02 | 0.00 | 0.01 | 2 | 2 | 33 | 42 |
| | | C3-75 | 3099 | 1809 | 37 | 1274 | 38 | 487061 | 0.00 | 0.00 | 0.00 | 3 | 3 | 22 | 24 |
| | | C4-75 | 2052 | 1851 | 75 | 946 | 43 | 398216 | 2.27 | 0.00 | 0.01 | 1 | 2 | 14 | 19 |
| | | C5-75 | 823 | 810 | 1 | 267 | 61 | 230460 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C6-75 | 3427 | 2012 | 39 | 1519 | 35 | 450414 | 2.85 | 0.00 | 2.85 | 48 | 48 | 107 | 243 |
| | | C7-75 | 2458 | 1649 | 21 | 619 | 46 | 437743 | 2.17 | 0.00 | 2.17 | 1 | 1 | 1 | 2 |
| | | C8-75 | 3002 | 2183 | 41 | 1049 | 37 | 487245 | 2.69 | 0.00 | 2.69 | 2 | 2 | 26 | 27 |
| | | C9-75 | 3550 | 2341 | 399 | 2597 | 31 | 327993 | 3.22 | 0.00 | 3.22 | 14 | 15 | 721 | 781 |
| | | C10-75 | 2856 | 1580 | 197 | 2350 | 32 | 526188 | 3.12 | 0.00 | 3.12 | 130 | 131 | 382 | 1854 |
| | | C11-75 | 1955 | 1193 | 23 | 645 | 47 | 443770 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 3 |
| | | C12-75 | 1801 | 1347 | 5 | 506 | 46 | 298162 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C13-75 | 2699 | 1302 | 161 | 1367 | 37 | 479166 | 2.70 | 0.00 | 2.70 | 3 | 4 | 59 | 67 |
| | | C14-75 | 2045 | 1333 | 21 | 626 | 47 | 301108 | 0.00 | 0.00 | 0.00 | 1 | 1 | 2 | 3 |
| | | C15-75 | 3003 | 1443 | 35 | 1117 | 36 | 460087 | 0.01 | 0.00 | 0.01 | 3 | 3 | 3 | 23 |
| | | C17-75 | 1492 | 835 | 3 | 404 | 50 | 348581 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C19-75 | 1497 | 664 | 1 | 390 | 57 | 245827 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C20-75 | 2323 | 1254 | 17 | 683 | 46 | 420825 | 0.00 | 0.00 | 0.00 | 1 | 1 | 2 | 2 |
| | | C22-75 | 3113 | 1524 | 199 | 1399 | 40 | 472040 | 2.50 | 0.00 | 2.49 | 3 | 3 | 19 | 105 |
| | | C23-75 | 2069 | 1548 | 25 | 983 | 35 | 379234 | 2.84 | 0.00 | 2.84 | 3 | 3 | 13 | 18 |
| | | C24-75 | 2446 | 1694 | 705 | 1620 | 37 | 425806 | 5.25 | 0.00 | 2.70 | 3 | 6 | 109 | 252 |
| | | C26-75 | 1957 | 1039 | 17 | 430 | 52 | 333692 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C28-75 | 1781 | 1452 | 5 | 628 | 44 | 293838 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C29-75 | 2495 | 2238 | 619 | 2097 | 32 | 543850 | 0.02 | 0.00 | 0.02 | 8 | 8 | 454 | 578 |
| | 6 | C0-75 | 1661 | 1565 | 15 | 462 | 47 | 356296 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C2-75 | 1906 | 1405 | 87 | 1072 | 40 | 643540 | 0.02 | 0.00 | 0.01 | 2 | 3 | 27 | 39 |
| | | C3-75 | 3099 | 1809 | 35 | 1281 | 38 | 487061 | 0.00 | 0.00 | 0.00 | 4 | 4 | 30 | 33 |
| | | C4-75 | 2052 | 1851 | 75 | 929 | 43 | 398216 | 2.27 | 0.00 | 0.01 | 2 | 2 | 16 | 20 |
| | | C5-75 | 823 | 810 | 1 | 267 | 61 | 230460 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C6-75 | 3427 | 2012 | 53 | 1557 | 35 | 450414 | 2.85 | 0.00 | 2.85 | 42 | 42 | 224 | 336 |
| | | C7-75 | 2458 | 1649 | 29 | 617 | 46 | 437743 | 2.17 | 0.00 | 2.17 | 1 | 1 | 1 | 2 |
| | | C8-75 | 3002 | 2183 | 19 | 1019 | 37 | 487245 | 2.69 | 0.00 | 2.69 | 2 | 2 | 10 | 14 |
| | | C9-75 | 3550 | 2341 | 103 | 2127 | 31 | 327993 | 3.22 | 0.00 | 3.22 | 17 | 18 | 246 | 339 |
| | | C10-75 | 2856 | 1580 | 73 | 2029 | 32 | 525115 | 3.11 | 0.00 | 3.11 | 2763 | 2764 | 2764 | 17162 |
| | | C11-75 | 1955 | 1193 | 23 | 648 | 47 | 443770 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 3 |
| | | C12-75 | 1801 | 1347 | 5 | 505 | 46 | 298162 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C13-75 | 2699 | 1302 | 369 | 1484 | 37 | 479166 | 2.70 | 0.00 | 2.70 | 5 | 5 | 88 | 151 |
| | | C14-75 | 2045 | 1333 | 15 | 620 | 47 | 301108 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 2 |
| | | C15-75 | 3007 | 1443 | 19 | 1061 | 36 | 460087 | 0.01 | 0.00 | 0.01 | 3 | 3 | 3 | 13 |
| | | C17-75 | 1492 | 835 | 5 | 411 | 50 | 348581 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C19-75 | 1497 | 664 | 1 | 390 | 57 | 245827 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C20-75 | 2323 | 1254 | 17 | 671 | 46 | 420825 | 0.00 | 0.00 | 0.00 | 1 | 1 | 2 | 2 |
| | | C22-75 | 3113 | 1524 | 141 | 1358 | 40 | 472040 | 2.50 | 0.00 | 2.49 | 3 | 4 | 53 | 84 |
| | | C23-75 | 2069 | 1548 | 35 | 981 | 35 | 379234 | 2.84 | 0.00 | 2.84 | 3 | 3 | 16 | 26 |
| | | C24-75 | 2446 | 1694 | 613 | 1572 | 37 | 425806 | 5.25 | 0.00 | 2.70 | 4 | 9 | 94 | 256 |
| | | C26-75 | 1957 | 1039 | 17 | 430 | 52 | 333692 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C28-75 | 1781 | 1452 | 5 | 622 | 44 | 293838 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C29-75 | 2495 | 2238 | 555 | 2050 | 32 | 543850 | 0.02 | 0.00 | 0.02 | 9 | 9 | 9 | 667 |

| Cluster size | Vehicle capacity | Cluster ID | Inbound edge # | Outbound edge # | Tree node # | Column # | Vehicle # | Total distance (m) | Optimality gap (%) | | Integrality gap (%) | Wall time (s) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | Root node soln. | Best feasible soln. | | RMP conv. | Root node soln. | Best feasible soln. | Total |
| 75 | 7 | C0-75 | 1661 | 1565 | 15 | 463 | 47 | 356296 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C2-75 | 1906 | 1405 | 79 | 1072 | 40 | 643540 | 0.02 | 0.00 | 0.01 | 3 | 3 | 30 | 39 |
| | | C3-75 | 3099 | 1809 | 21 | 1238 | 38 | 487061 | 0.00 | 0.00 | 0.00 | 4 | 4 | 4 | 20 |
| | | C4-75 | 2052 | 1851 | 139 | 962 | 43 | 398216 | 2.27 | 0.00 | 0.01 | 1 | 2 | 33 | 33 |
| | | C5-75 | 823 | 810 | 1 | 267 | 61 | 230460 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C6-75 | 3427 | 2012 | 33 | 1515 | 35 | 450414 | 2.85 | 0.00 | 2.85 | 48 | 48 | 186 | 247 |
| | | C7-75 | 2458 | 1649 | 31 | 616 | 46 | 437743 | 2.17 | 0.00 | 2.17 | 1 | 1 | 1 | 3 |
| | | C8-75 | 3002 | 2183 | 23 | 1025 | 37 | 487245 | 2.69 | 0.00 | 2.69 | 2 | 2 | 9 | 18 |
| | | C9-75 | 3550 | 2341 | 189 | 2347 | 31 | 327993 | 3.22 | 0.00 | 3.22 | 20 | 20 | 482 | 594 |
| | | C10-75 | 2856 | 1580 | 2 | 1459 | 32 | 525115 | 3.12 | 3.12 | 3.12 | 39153 | 39154 | 39154 | 43200 |
| | | C11-75 | 1955 | 1193 | 24 | 650 | 47 | 443770 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 3 |
| | | C12-75 | 1801 | 1347 | 5 | 502 | 46 | 298162 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C13-75 | 2699 | 1302 | 337 | 1470 | 37 | 479166 | 2.70 | 0.00 | 2.70 | 2 | 3 | 78 | 105 |
| | | C14-75 | 2045 | 1332 | 19 | 619 | 47 | 301108 | 0.00 | 0.00 | 0.00 | 1 | 1 | 2 | 2 |
| | | C15-75 | 3007 | 1443 | 25 | 1089 | 36 | 460087 | 0.01 | 0.00 | 0.01 | 2 | 3 | 3 | 17 |
| | | C17-75 | 1492 | 835 | 7 | 411 | 50 | 348581 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C19-75 | 1497 | 664 | 1 | 390 | 57 | 245827 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C20-75 | 2323 | 1254 | 11 | 672 | 46 | 420825 | 0.00 | 0.00 | 0.00 | 1 | 1 | 2 | 2 |
| | | C22-75 | 3113 | 1524 | 161 | 1382 | 40 | 472040 | 2.50 | 0.00 | 2.49 | 3 | 3 | 52 | 79 |
| | | C23-75 | 2069 | 1548 | 23 | 940 | 35 | 379234 | 2.84 | 0.00 | 2.84 | 3 | 3 | 12 | 19 |
| | | C24-75 | 2446 | 1694 | 745 | 1666 | 37 | 425806 | 5.25 | 0.00 | 2.70 | 3 | 6 | 148 | 329 |
| | | C26-75 | 1957 | 1039 | 17 | 430 | 52 | 333692 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C28-75 | 1781 | 1452 | 5 | 621 | 44 | 293838 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C29-75 | 2495 | 2238 | 523 | 2052 | 32 | 543850 | 0.02 | 0.00 | 0.02 | 7 | 8 | 8 | 619 |
| | 8 | C0-75 | 1661 | 1565 | 15 | 461 | 47 | 356296 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C2-75 | 1906 | 1405 | 75 | 1047 | 40 | 643540 | 0.02 | 0.00 | 0.01 | 2 | 2 | 26 | 36 |
| | | C3-75 | 3099 | 1809 | 35 | 1268 | 38 | 487061 | 0.00 | 0.00 | 0.00 | 4 | 4 | 30 | 33 |
| | | C4-75 | 2052 | 1851 | 101 | 928 | 43 | 398216 | 0.01 | 0.00 | 0.01 | 2 | 2 | 27 | 28 |
| | | C5-75 | 823 | 810 | 1 | 267 | 61 | 230460 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C6-75 | 3427 | 2012 | 62 | 1584 | 35 | 450414 | 5.54 | 0.00 | 2.85 | 39 | 40 | 266 | 362 |
| | | C7-75 | 2458 | 1649 | 31 | 618 | 46 | 437743 | 2.17 | 0.00 | 2.17 | 1 | 1 | 1 | 2 |
| | | C8-75 | 3002 | 2183 | 49 | 1064 | 37 | 487245 | 2.69 | 0.00 | 2.69 | 3 | 4 | 9 | 81 |
| | | C9-75 | 3550 | 2341 | 105 | 2199 | 31 | 327993 | 3.22 | 0.00 | 3.22 | 26 | 27 | 300 | 398 |
| | | C10-75 | 2856 | 1580 | 1 | 1425 | 32 | 525207 | 3.12 | 3.12 | 3.12 | 84928 | 84928 | 84928 | 84928 |
| | | C11-75 | 1955 | 1193 | 24 | 641 | 47 | 443770 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 3 |
| | | C12-75 | 1801 | 1347 | 5 | 508 | 46 | 298162 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C13-75 | 2699 | 1302 | 427 | 1513 | 37 | 479166 | 2.70 | 0.00 | 2.70 | 4 | 4 | 77 | 144 |
| | | C14-75 | 2045 | 1332 | 17 | 624 | 47 | 301108 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 2 |
| | | C15-75 | 3005 | 1443 | 21 | 1093 | 36 | 460087 | 0.01 | 0.00 | 0.01 | 3 | 3 | 3 | 16 |
| | | C17-75 | 1492 | 835 | 7 | 410 | 50 | 348581 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C19-75 | 1497 | 664 | 1 | 390 | 57 | 245827 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C20-75 | 2323 | 1254 | 11 | 671 | 46 | 420825 | 0.00 | 0.00 | 0.00 | 1 | 1 | 2 | 2 |
| | | C22-75 | 3113 | 1524 | 161 | 1357 | 40 | 472040 | 2.50 | 0.00 | 2.49 | 3 | 4 | 65 | 103 |
| | | C23-75 | 2069 | 1548 | 25 | 959 | 35 | 379234 | 2.84 | 0.00 | 2.84 | 4 | 4 | 16 | 22 |
| | | C24-75 | 2446 | 1694 | 677 | 1567 | 37 | 425806 | 2.70 | 0.00 | 2.70 | 3 | 3 | 83 | 265 |
| | | C26-75 | 1957 | 1039 | 15 | 430 | 52 | 333692 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C28-75 | 1781 | 1452 | 5 | 620 | 44 | 293838 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 1 |
| | | C29-75 | 2495 | 2238 | 573 | 2163 | 32 | 543850 | 0.02 | 0.00 | 0.02 | 9 | 9 | 353 | 625 |
| 100 | 4 | C0-100 | 3488 | 1930 | 45 | 777 | 63 | 488119 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 5 |
| | | C1-100 | 1825 | 1642 | 5 | 429 | 75 | 331497 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 3 |
| | | C2-100 | 5540 | 3106 | 1445 | 2652 | 46 | 596824 | 2.17 | 0.00 | 2.17 | 6 | 7 | 995 | 1397 |
| | | C3-100 | 5383 | 3127 | 823 | 2794 | 46 | 586434 | 2.17 | 0.00 | 2.16 | 8 | 9 | 558 | 695 |
| | | C4-100 | 4211 | 2787 | 50653 | 6120 | 44 | 558323 | 2.27 | 0.00 | 2.26 | 8 | 9 | 730 | 31745 |
| | | C5-100 | 1960 | 1452 | 39 | 575 | 70 | 366565 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 4 |
| | | C6-100 | 4708 | 3393 | 33 | 1805 | 47 | 724844 | 0.00 | 0.00 | 0.00 | 8 | 8 | 39 | 44 |
| | | C7-100 | 4120 | 2205 | 23 | 1227 | 53 | 661648 | 0.00 | 0.00 | 0.00 | 3 | 3 | 3 | 10 |
| | | C8-100 | 4952 | 2656 | 3 | 1285 | 51 | 609120 | 0.00 | 0.00 | 0.00 | 3 | 3 | 3 | 4 |
| | | C9-100 | 5664 | 3572 | 2316 | 6235 | 40 | 527217 | 2.50 | 0.00 | 2.49 | 57 | 59 | 7410 | 14795 |
| | | C10-100 | 2995 | 2611 | 147 | 980 | 58 | 426386 | 1.72 | 0.00 | 1.72 | 2 | 2 | 14 | 16 |
| | | C11-100 | 4606 | 2964 | 61 | 1506 | 51 | 427829 | 0.00 | 0.00 | 0.00 | 3 | 4 | 14 | 32 |
| | | C12-100 | 2863 | 2459 | 1 | 794 | 62 | 413012 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 2 |
| | | C13-100 | 3232 | 2441 | 17 | 755 | 61 | 483087 | 0.00 | 0.00 | 0.00 | 2 | 2 | 3 | 3 |
| | | C14-100 | 4335 | 3403 | 2695 | 3757 | 40 | 693825 | 4.85 | 0.00 | 2.49 | 15 | 1297 | 3093 | 6348 |
| | | C15-100 | 3711 | 2642 | 26493 | 3217 | 48 | 790005 | 4.16 | 0.00 | 4.16 | 7 | 8 | 2068 | 26100 |
| | | C16-100 | 3278 | 2302 | 21 | 1085 | 55 | 627423 | 1.81 | 0.00 | 1.81 | 3 | 3 | 5 | 8 |
| | | C17-100 | 3413 | 1603 | 7 | 881 | 60 | 640945 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 3 |
| | | C18-100 | 3778 | 2518 | 81 | 872 | 58 | 681240 | 1.72 | 0.00 | 1.72 | 2 | 2 | 2 | 10 |
| | | C19-100 | 3722 | 3405 | 353 | 1656 | 51 | 503252 | 1.96 | 0.00 | 1.96 | 4 | 4 | 64 | 192 |
| | | C20-100 | 4377 | 2653 | 1863 | 2708 | 46 | 569724 | 2.17 | 0.00 | 2.17 | 7 | 7 | 1135 | 1144 |
| | | C21-100 | 2597 | 1223 | 1 | 524 | 78 | 328216 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 2 |

| Cluster size | Vehicle capacity | Cluster ID | Inbound edge # | Outbound edge # | Tree node # | Column # | Vehicle # | Total distance (m) | Optimality gap (%) | | Integrality gap (%) | Wall time (s) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | Root node soln. | Best feasible soln. | | RMP conv. | Root node soln. | Best feasible soln. | Total |
| 100 | 5 | C0-100 | 3488 | 1930 | 43 | 772 | 63 | 488119 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 5 |
| | | C1-100 | 1825 | 1642 | 5 | 429 | 75 | 331497 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 2 |
| | | C2-100 | 5540 | 3106 | 1513 | 2637 | 46 | 596658 | 2.17 | 0.00 | 2.17 | 9 | 10 | 1269 | 1942 |
| | | C3-100 | 5383 | 3127 | 191 | 2449 | 46 | 584077 | 2.17 | 0.00 | 2.16 | 60 | 61 | 860 | 950 |
| | | C4-100 | 4209 | 2787 | 67 | 2111 | 44 | 552862 | 2.26 | 0.00 | 2.26 | 18 | 27 | 141 | 183 |
| | | C5-100 | 1964 | 1452 | 47 | 586 | 70 | 366565 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 4 |
| | | C6-100 | 4708 | 3393 | 29 | 1807 | 47 | 724844 | 0.00 | 0.00 | 0.00 | 8 | 8 | 26 | 54 |
| | | C7-100 | 4120 | 2205 | 27 | 1234 | 53 | 661528 | 0.00 | 0.00 | 0.00 | 4 | 4 | 4 | 13 |
| | | C8-100 | 4952 | 2656 | 3 | 1287 | 51 | 609038 | 0.00 | 0.00 | 0.00 | 3 | 3 | 3 | 5 |
| | | C9-100 | 5559 | 3570 | 459 | 4860 | 40 | 521285 | 2.49 | 2.49 | 2.49 | 798 | 799 | 29422 | 43304 |
| | | C10-100 | 2995 | 2611 | 169 | 971 | 58 | 426386 | 1.72 | 0.00 | 1.72 | 2 | 2 | 2 | 17 |
| | | C11-100 | 4606 | 2964 | 53 | 1462 | 51 | 427829 | 0.00 | 0.00 | 0.00 | 4 | 5 | 18 | 38 |
| | | C12-100 | 2863 | 2459 | 1 | 787 | 62 | 413012 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 2 |
| | | C13-100 | 3232 | 2441 | 17 | 758 | 61 | 483087 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 3 |
| | | C14-100 | 4335 | 3403 | 1067 | 3293 | 40 | 690964 | 2.49 | 2.49 | 2.49 | 70 | 72 | 39918 | 43218 |
| | | C15-100 | 3711 | 2642 | 11238 | 4146 | 48 | 788340 | 4.16 | 4.16 | 4.16 | 22 | 22 | 4167 | 43205 |
| | | C16-100 | 3278 | 2302 | 65 | 1151 | 55 | 626787 | 1.81 | 0.00 | 1.81 | 2 | 2 | 14 | 24 |
| | | C17-100 | 3413 | 1603 | 19 | 902 | 60 | 640945 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 5 |
| | | C18-100 | 3778 | 2518 | 65 | 868 | 58 | 681240 | 1.72 | 0.00 | 1.72 | 2 | 3 | 3 | 9 |
| | | C19-100 | 3722 | 3405 | 633 | 1695 | 51 | 503252 | 1.96 | 0.00 | 1.96 | 4 | 4 | 263 | 359 |
| | | C20-100 | 4377 | 2653 | 1785 | 2948 | 46 | 568797 | 2.17 | 0.00 | 2.17 | 12 | 13 | 1765 | 1815 |
| | | C21-100 | 2597 | 1223 | 1 | 525 | 78 | 328216 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 2 |
| | 6 | C0-100 | 3488 | 1931 | 59 | 782 | 63 | 488119 | 0.00 | 0.00 | 0.00 | 2 | 3 | 3 | 7 |
| | | C1-100 | 1825 | 1642 | 5 | 429 | 75 | 331497 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 2 |
| | | C2-100 | 5540 | 3106 | 1439 | 2614 | 46 | 596658 | 2.17 | 0.00 | 2.17 | 10 | 11 | 1618 | 2309 |
| | | C3-100 | 5383 | 3127 | 415 | 2690 | 46 | 584048 | 2.17 | 0.00 | 2.16 | 76 | 77 | 1791 | 1841 |
| | | C4-100 | 4211 | 2787 | 309 | 2656 | 44 | 552862 | 2.26 | 0.00 | 2.26 | 21 | 24 | 178 | 773 |
| | | C5-100 | 1959 | 1452 | 43 | 583 | 70 | 366565 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 4 |
| | | C6-100 | 4708 | 3393 | 27 | 1821 | 47 | 724844 | 0.00 | 0.00 | 0.00 | 11 | 11 | 30 | 85 |
| | | C7-100 | 4120 | 2205 | 29 | 1243 | 53 | 661528 | 0.00 | 0.00 | 0.00 | 4 | 4 | 4 | 15 |
| | | C8-100 | 4952 | 2656 | 3 | 1277 | 51 | 609038 | 0.00 | 0.00 | 0.00 | 3 | 3 | 3 | 4 |
| | | C9-100 | 5664 | 3572 | 35 | 2997 | 40 | 518571 | 2.49 | 2.49 | 2.49 | 10273 | 10275 | 10275 | 43211 |
| | | C10-100 | 2995 | 2611 | 149 | 982 | 58 | 426386 | 1.72 | 0.00 | 1.72 | 2 | 2 | 16 | 17 |
| | | C11-100 | 4606 | 2964 | 45 | 1453 | 51 | 427829 | 0.00 | 0.00 | 0.00 | 4 | 4 | 18 | 40 |
| | | C12-100 | 2863 | 2459 | 1 | 790 | 62 | 413012 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 2 |
| | | C13-100 | 3232 | 2441 | 17 | 760 | 61 | 483087 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 3 |
| | | C14-100 | 4335 | 3403 | 532 | 3161 | 40 | 693512 | 2.50 | 2.50 | 2.50 | 144 | 148 | 21052 | 43203 |
| | | C15-100 | 3711 | 2642 | 3810 | 3876 | 48 | 788340 | 4.16 | 4.16 | 4.16 | 142 | 142 | 15775 | 43217 |
| | | C16-100 | 3278 | 2302 | 69 | 1151 | 55 | 626787 | 1.81 | 0.00 | 1.81 | 3 | 3 | 3 | 41 |
| | | C17-100 | 3411 | 1603 | 9 | 881 | 60 | 640945 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 3 |
| | | C18-100 | 3778 | 2518 | 59 | 867 | 58 | 681240 | 1.72 | 0.00 | 1.72 | 2 | 2 | 2 | 8 |
| | | C19-100 | 3722 | 3405 | 1364 | 1761 | 51 | 503252 | 1.96 | 0.00 | 1.96 | 5 | 5 | 696 | 977 |
| | | C20-100 | 4377 | 2653 | 2993 | 3190 | 46 | 568797 | 2.17 | 0.00 | 2.17 | 21 | 22 | 3242 | 3252 |
| | | C21-100 | 2597 | 1223 | 1 | 527 | 78 | 328216 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 2 |
| | 7 | C0-100 | 3488 | 1931 | 65 | 783 | 63 | 488119 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 6 |
| | | C1-100 | 1825 | 1642 | 5 | 429 | 75 | 331497 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 2 |
| | | C2-100 | 5540 | 3106 | 1613 | 2710 | 46 | 596658 | 2.17 | 0.00 | 2.17 | 11 | 12 | 1811 | 2822 |
| | | C3-100 | 5383 | 3127 | 247 | 2572 | 46 | 584048 | 2.17 | 0.00 | 2.16 | 68 | 69 | 1250 | 1297 |
| | | C4-100 | 4209 | 2787 | 87 | 2212 | 44 | 552334 | 2.26 | 0.00 | 2.26 | 13 | 15 | 220 | 240 |
| | | C5-100 | 1959 | 1452 | 35 | 583 | 70 | 366565 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 4 |
| | | C6-100 | 4708 | 3393 | 27 | 1807 | 47 | 724844 | 0.00 | 0.00 | 0.00 | 11 | 11 | 29 | 82 |
| | | C7-100 | 4120 | 2205 | 27 | 1209 | 53 | 661528 | 0.00 | 0.00 | 0.00 | 3 | 4 | 4 | 13 |
| | | C8-100 | 4952 | 2656 | 3 | 1270 | 51 | 609038 | 0.00 | 0.00 | 0.00 | 3 | 3 | 3 | 4 |
| | | C9-100 | 5664 | 3572 | 1 | 2482 | 40 | 520597 | 2.50 | 2.50 | 2.50 | 43846 | 43847 | 43847 | 43847 |
| | | C10-100 | 2995 | 2611 | 155 | 981 | 58 | 426386 | 1.72 | 0.00 | 1.72 | 2 | 2 | 16 | 17 |
| | | C11-100 | 4606 | 2964 | 57 | 1456 | 51 | 427829 | 0.00 | 0.00 | 0.00 | 5 | 5 | 29 | 49 |
| | | C12-100 | 2863 | 2459 | 1 | 793 | 62 | 413012 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 2 |
| | | C13-100 | 3232 | 2441 | 17 | 757 | 61 | 483087 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 3 |
| | | C14-100 | 4335 | 3403 | 369 | 2927 | 40 | 692956 | 2.49 | 2.49 | 2.49 | 240 | 242 | 242 | 43244 |
| | | C15-100 | 3711 | 2642 | 993 | 2445 | 48 | 789091 | 4.16 | 4.16 | 4.16 | 942 | 942 | 942 | 43215 |
| | | C16-100 | 3278 | 2302 | 63 | 1131 | 55 | 626787 | 1.81 | 0.00 | 1.81 | 4 | 4 | 28 | 53 |
| | | C17-100 | 3413 | 1603 | 9 | 875 | 60 | 640945 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 3 |
| | | C18-100 | 3778 | 2518 | 81 | 872 | 58 | 681240 | 1.72 | 0.00 | 1.72 | 2 | 2 | 2 | 10 |
| | | C19-100 | 3722 | 3405 | 639 | 1714 | 51 | 503252 | 1.96 | 0.00 | 1.96 | 5 | 6 | 306 | 407 |
| | | C20-100 | 4377 | 2653 | 1669 | 2890 | 46 | 568797 | 2.17 | 0.00 | 2.17 | 16 | 17 | 1180 | 2009 |
| | | C21-100 | 2597 | 1223 | 1 | 525 | 78 | 328216 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 2 |
| | 8 | C0-100 | 3488 | 1931 | 65 | 784 | 63 | 488119 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 7 |
| | | C1-100 | 1825 | 1642 | 5 | 429 | 75 | 331497 | 0.00 | 0.00 | 0.00 | 1 | 1 | 1 | 2 |
| | | C2-100 | 5540 | 3106 | 1477 | 2619 | 46 | 596658 | 2.17 | 0.00 | 2.17 | 13 | 13 | 1790 | 2605 |
| | | C3-100 | 5383 | 3127 | 71 | 2209 | 46 | 584048 | 2.17 | 0.00 | 2.16 | 90 | 91 | 389 | 518 |
| | | C4-100 | 4210 | 2787 | 71 | 2176 | 44 | 552334 | 2.26 | 0.00 | 2.26 | 13 | 15 | 253 | 271 |
| | | C5-100 | 1964 | 1452 | 53 | 575 | 70 | 366565 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 5 |
| | | C6-100 | 4708 | 3393 | 27 | 1822 | 47 | 724844 | 0.00 | 0.00 | 0.00 | 11 | 12 | 29 | 87 |
| | | C7-100 | 4120 | 2205 | 29 | 1250 | 53 | 661528 | 0.00 | 0.00 | 0.00 | 3 | 4 | 4 | 15 |
| | | C8-100 | 4952 | 2656 | 3 | 1273 | 51 | 609038 | 0.00 | 0.00 | 0.00 | 3 | 3 | 3 | 5 |
| | | C9-100 | 5664 | 3572 | 4 | 2577 | 40 | 517627 | 2.49 | 2.49 | 2.49 | 34816 | 34818 | 34818 | 43200 |
| | | C10-100 | 2995 | 2611 | 159 | 976 | 58 | 426386 | 1.72 | 0.00 | 1.72 | 2 | 2 | 16 | 18 |
| | | C11-100 | 4606 | 2964 | 53 | 1457 | 51 | 427829 | 0.00 | 0.00 | 0.00 | 4 | 5 | 19 | 45 |
| | | C12-100 | 2863 | 2459 | 1 | 789 | 62 | 413012 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 2 |
| | | C13-100 | 3232 | 2441 | 13 | 760 | 61 | 483087 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 3 |
| | | C14-100 | 4060 | 3403 | 460 | 3150 | 40 | 693850 | 2.50 | 2.50 | 2.50 | 196 | 198 | 33298 | 43354 |
| | | C15-100 | 3711 | 2642 | 321 | 2353 | 48 | 789091 | 4.16 | 4.16 | 4.16 | 2597 | 2598 | 2598 | 44157 |
| | | C16-100 | 3278 | 2302 | 29 | 1116 | 55 | 626787 | 1.81 | 0.00 | 1.81 | 4 | 4 | 13 | 32 |
| | | C17-100 | 3413 | 1603 | 9 | 873 | 60 | 640945 | 0.00 | 0.00 | 0.00 | 2 | 2 | 2 | 3 |
| | | C18-100 | 3778 | 2518 | 71 | 871 | 58 | 681240 | 1.72 | 0.00 | 1.72 | 2 | 2 | 2 | 9 |
| | | C19-100 | 3722 | 3405 | 719 | 1709 | 51 | 503252 | 1.96 | 0.00 | 1.96 | 5 | 5 | 394 | 487 |
| | | C20-100 | 4377 | 2653 | 1760 | 2899 | 46 | 568797 | 2.17 | 0.00 | 2.17 | 21 | 22 | 1737 | 2192 |
| | | C21-100 | 2597 | 1223 | 1 | 524 | 78 | 328216 | 0.00 | 0.00 | 0.00 | 1 | 2 | 2 | 2 |

Tables A.3 and A.4 summarize results of the cluster size scaling experiments for the REA and the BPA respectively. They list the same quantities as those listed in Tables A.1 and A.2 respectively.

Table A.3: Results of REA Scalability with Increasing Cluster Size ($K = 4$, $\Delta = 10$ mins, $R = 50\%$).

| Cluster size | Cluster ID | Column # | Vehicle # | Total distance (m) | Optimality gap (%) | Wall time (s) | |
|---|---|---|---|---|---|---|---|
| | | | | | | Route enumeration | Total |
| 200 | C0-200 | 2286 | 129 | 702670 | 0.00 | 618 | 618 |
| | C1-200 | 10955 | 84 | 1122469 | 0.00 | 653 | 654 |
| | C2-200 | 15149 | 85 | 1067503 | 0.00 | 646 | 2361 |
| | C3-200 | 3101 | 113 | 824730 | 0.00 | 629 | 630 |
| | C4-200 | 2627 | 114 | 684981 | 0.00 | 632 | 878 |
| | C5-200 | 22061 | 80 | 937514 | 0.00 | 660 | 744 |
| | C6-200 | 11844 | 86 | 1136870 | 0.00 | 673 | 680 |
| | C7-200 | 16235 | 76 | 1365475 | 0.00 | 668 | 1165 |
| | C8-200 | 27913 | 78 | 923290 | 0.49 | 684 | 43211 |
| | C9-200 | 5289 | 99 | 1055393 | 0.00 | 655 | 702 |
| | C10-200 | 7790 | 87 | 1249784 | 0.00 | 676 | 8731 |
| 300 | C0-300 | 33262 | 118 | 1652972 | 0.16 | 3301 | 43206 |
| | C1-300 | 32536 | 119 | 1928887 | 0.44 | 2796 | 43203 |
| | C2-300 | 7994 | 150 | 1096341 | 0.00 | 3249 | 4367 |
| | C3-300 | 36568 | 113 | 1511806 | 0.00 | 3499 | 3650 |
| | C4-300 | 15394 | 134 | 1477823 | 0.13 | 3198 | 43205 |
| | C5-300 | 22072 | 124 | 1773852 | 0.00 | 3342 | 4325 |
| | C6-300 | 33541 | 119 | 1927882 | 0.35 | 3191 | 43205 |
| | C7-300 | 6554 | 157 | 1105570 | 0.00 | 3152 | 3158 |
| | C8-300 | 53120 | 114 | 1418494 | 0.20 | 3129 | 43204 |
| | C9-300 | 30568 | 114 | 1911085 | 0.17 | 2258 | 43203 |
| | C10-300 | 6370 | 156 | 1085481 | 0.00 | 2736 | 2776 |
| | C11-300 | 34630 | 118 | 1903034 | 0.37 | 3037 | 43204 |
| | C12-300 | 7137 | 153 | 1067479 | 0.00 | 2339 | 2632 |
| 400 | C0-400 | 28968 | 180 | 1617748 | 0.00 | 8897 | 35048 |
| | C1-400 | 52194 | 145 | 1901974 | 0.19 | 6972 | 43203 |
| | C2-400 | 28025 | 173 | 1627074 | 0.00 | 10060 | 18906 |
| | C3-400 | 41012 | 152 | 2114911 | 0.20 | 7842 | 43203 |
| | C4-400 | 26314 | 182 | 1687079 | 0.00 | 10447 | 10531 |
| | C5-400 | 53948 | 151 | 1988361 | 0.00 | 7043 | 12685 |
| | C6-400 | 26109 | 173 | 1640438 | 0.00 | 10716 | 10986 |
| | C7-400 | 68190 | 145 | 1887086 | 0.32 | 7117 | 43202 |

Table A.4: Results of BPA Scalability with Increasing Cluster Size ($K = 4$, $\Delta = 10$ mins, $R = 50\%$).

| Cluster size | Cluster ID | Inbound edge # | Outbound edge # | Tree node # | Column # | Vehicle # | Total distance (m) | Optimality gap (%) | | Integrality gap (%) | Wall time (s) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Root node soln. | Best feasible soln. | | RMP conv. | Root node soln. | Best feasible soln. | Total |
| 200 | C0-200 | 10281 | 5900 | 113 | 1747 | 129 | 702670 | 1.55 | 0.00 | 1.55 | 13 | 13 | 35 | 40 |
| | C1-200 | 17096 | 10087 | 5870 | 6121 | 84 | 1122544 | 3.50 | 2.37 | 2.37 | 59 | 96 | 6951 | 43201 |
| | C2-200 | 13922 | 9341 | 7479 | 7506 | 85 | 1067640 | 2.35 | 2.34 | 2.34 | 55 | 74 | 6263 | 43203 |
| | C3-200 | 10551 | 7643 | 99 | 2272 | 113 | 824730 | 0.00 | 0.00 | 0.00 | 19 | 20 | 20 | 78 |
| | C4-200 | 9806 | 8163 | 5901 | 2616 | 114 | 684981 | 1.75 | 0.00 | 1.75 | 17 | 31 | 2896 | 4149 |
| | C5-200 | 18385 | 10756 | 3695 | 9115 | 80 | 937514 | 1.26 | 1.25 | 1.25 | 111 | 121 | 14244 | 43203 |
| | C6-200 | 16047 | 9533 | 13563 | 9572 | 86 | 1136870 | 2.29 | 1.16 | 1.16 | 45 | 202 | 14218 | 43201 |
| | C7-200 | 15672 | 11696 | 2279 | 8993 | 76 | 1367209 | 3.88 | 2.62 | 2.62 | 140 | 6148 | 23012 | 43213 |
| | C8-200 | 16483 | 12874 | 1 | 6027 | 79 | 928401 | 2.53 | 2.53 | 2.53 | 319 | 43209 | 43209 | 43209 |
| | C9-200 | 14587 | 9841 | 15783 | 4425 | 99 | 1055393 | 1.01 | 0.00 | 1.00 | 25 | 37 | 6671 | 20722 |
| | C10-200 | 15459 | 11600 | 4244 | 6328 | 87 | 1249965 | 2.29 | 2.29 | 2.29 | 45 | 150 | 9471 | 43202 |
| 300 | C0-300 | 38471 | 31409 | 242 | 11624 | 118 | 1665487 | 0.85 | 0.85 | 0.85 | 900 | 1018 | 1018 | 43265 |
| | C1-300 | 36931 | 22949 | 496 | 12533 | 119 | 1938007 | 2.52 | 2.52 | 2.52 | 944 | 4151 | 4151 | 43229 |
| | C2-300 | 26167 | 20901 | 11208 | 6629 | 150 | 1096341 | 1.33 | 1.33 | 1.33 | 81 | 124 | 7967 | 43200 |
| | C3-300 | 35614 | 23695 | 762 | 13192 | 114 | 1510444 | 1.75 | 1.75 | 1.75 | 426 | 12864 | 12864 | 43212 |
| | C4-300 | 30651 | 26297 | 1091 | 8757 | 134 | 1478291 | 1.49 | 1.49 | 1.49 | 278 | 519 | 39914 | 43243 |
| | C5-300 | 32453 | 25677 | 400 | 9653 | 125 | 1772096 | 0.80 | 0.80 | 0.80 | 622 | 14460 | 14460 | 43259 |
| | C6-300 | 35954 | 20702 | 483 | 12056 | 119 | 1934456 | 2.52 | 2.52 | 2.52 | 734 | 1914 | 1914 | 43218 |
| | C7-300 | 24094 | 19880 | 3185 | 5592 | 157 | 1105570 | 1.26 | 0.00 | 0.64 | 70 | 1027 | 3841 | 10016 |
| | C8-300 | 38059 | 22799 | 255 | 12441 | 115 | 1423615 | 2.60 | 2.60 | 2.60 | 480 | 29027 | 29027 | 43201 |
| | C9-300 | 33512 | 23617 | 779 | 13090 | 114 | 1934405 | 2.64 | 2.64 | 2.64 | 515 | 666 | 666 | 43252 |
| | C10-300 | 22339 | 16909 | 23037 | 5367 | 156 | 1085482 | 0.64 | 0.64 | 0.64 | 63 | 70 | 7449 | 43202 |
| | C11-300 | 34544 | 21130 | 698 | 12376 | 118 | 1910713 | 3.38 | 3.38 | 3.38 | 568 | 988 | 988 | 43227 |
| | C12-300 | 24520 | 17238 | 12001 | 6091 | 153 | 1067479 | 0.65 | 0.65 | 0.65 | 88 | 124 | 4043 | 43200 |
| 400 | C0-400 | 49218 | 39758 | 1 | 10357 | 181 | 1613436 | 1.65 | 1.65 | 1.65 | 528 | 43205 | 43205 | 43205 |
| | C1-400 | 53747 | 35507 | 1 | 13136 | 147 | 1926300 | 2.72 | 2.72 | 2.72 | 751 | 43207 | 43207 | 43207 |
| | C2-400 | 54300 | 35489 | 675 | 12751 | 173 | 1631283 | 1.16 | 1.16 | 1.16 | 406 | 540 | 540 | 43223 |
| | C3-400 | 55542 | 42004 | 1 | 12834 | 153 | 2124836 | 1.96 | 1.96 | 1.96 | 1596 | 43204 | 43204 | 43204 |
| | C4-400 | 49195 | 31033 | 3523 | 11881 | 182 | 1688282 | 1.10 | 1.10 | 1.10 | 222 | 239 | 25641 | 43201 |
| | C5-400 | 57633 | 34496 | 1 | 13557 | 152 | 1983300 | 1.97 | 1.97 | 1.97 | 684 | 43204 | 43204 | 43204 |
| | C6-400 | 46155 | 30320 | 3708 | 12494 | 173 | 1640472 | 1.16 | 1.15 | 1.15 | 240 | 289 | 11566 | 43210 |
| | C7-400 | 61393 | 38346 | 1 | 14833 | 146 | 1886534 | 2.73 | 2.73 | 2.73 | 960 | 43206 | 43206 | 43206 |

Finally, Table A.5 shows results of the experiments which investigate the efficiency of the root-node heuristic. The first two columns show the cluster size and ID of each problem instance. The next set of five columns list the results of the heuristic which enforces forbidden paths. They show the number of columns generated, the resulting vehicle count, its optimality gap, and the times spent on solving the RMP and its MIP. The next set of six columns display the same information for the heuristic which relaxes forbidden paths, with an additional column showing the number of infeasible columns it generated.

Table A.5: Results of Root-Node Heuristics with $t_{\text{RMP}} = 8$ mins and $t_{\text{MIP}} = 2$ mins ($K = 4$, $\Delta = 10$ mins, $R = 50\%$).

| Cluster size | Cluster ID | Enforce forbidden paths | | | | | Relax forbidden paths | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Column # | Vehicle # | Optimality gap (%) | Wall time (s) | | Column # | Infeasible column # | Vehicle # | Optimality gap (%) | Wall time (s) | |
| | | | | | RMP | MIP | | | | | RMP | MIP |
| 100 | C0-100 | 720 | 63 | 0.00 | 2 | 0 | 720 | 0 | 63 | 0.00 | 2 | 0 |
| | C1-100 | 422 | 75 | 0.00 | 2 | 0 | 423 | 0 | 75 | 0.00 | 2 | 0 |
| | C2-100 | 1828 | 46 | 2.17 | 5 | 0 | 1792 | 6 | 46 | 4.35 | 4 | 0 |
| | C3-100 | 1735 | 46 | 2.17 | 4 | 0 | 1729 | 6 | 46 | 2.17 | 4 | 0 |
| | C4-100 | 1653 | 44 | 2.27 | 6 | 5 | 1624 | 4 | 44 | 2.27 | 5 | 1 |
| | C5-100 | 537 | 70 | 0.00 | 1 | 0 | 541 | 0 | 70 | 0.00 | 2 | 0 |
| | C6-100 | 1637 | 47 | 0.00 | 5 | 0 | 1628 | 3 | 47 | 0.00 | 4 | 0 |
| | C7-100 | 1130 | 53 | 0.00 | 2 | 0 | 1123 | 5 | 53 | 0.00 | 2 | 0 |
| | C8-100 | 1198 | 52 | 1.92 | 2 | 0 | 1184 | 3 | 51 | 0.00 | 2 | 0 |
| | C9-100 | 2360 | 41 | 4.88 | 29 | 10 | 2467 | 10 | 41 | 4.88 | 37 | 2 |
| | C10-100 | 891 | 58 | 1.72 | 2 | 0 | 895 | 1 | 58 | 1.72 | 2 | 0 |
| | C11-100 | 1272 | 51 | 0.00 | 3 | 0 | 1252 | 0 | 51 | 0.00 | 2 | 0 |
| | C12-100 | 767 | 62 | 0.00 | 2 | 0 | 776 | 2 | 62 | 0.00 | 2 | 0 |
| | C13-100 | 708 | 61 | 0.00 | 2 | 0 | 708 | 0 | 61 | 0.00 | 1 | 0 |
| | C14-100 | 2003 | 41 | 4.88 | 11 | 2 | 1994 | 12 | 41 | 4.88 | 8 | 2 |
| | C15-100 | 1548 | 48 | 4.17 | 3 | 0 | 1580 | 9 | 48 | 6.25 | 4 | 0 |
| | C16-100 | 992 | 55 | 1.82 | 2 | 0 | 988 | 2 | 55 | 1.82 | 2 | 0 |
| | C17-100 | 817 | 60 | 0.00 | 2 | 0 | 810 | 3 | 60 | 0.00 | 2 | 0 |
| | C18-100 | 808 | 58 | 1.72 | 2 | 0 | 808 | 0 | 58 | 1.72 | 1 | 0 |
| | C19-100 | 1337 | 51 | 1.96 | 3 | 0 | 1334 | 9 | 52 | 3.85 | 3 | 0 |
| | C20-100 | 1672 | 46 | 2.17 | 4 | 1 | 1643 | 2 | 46 | 2.17 | 4 | 1 |
| | C21-100 | 500 | 78 | 0.00 | 2 | 0 | 503 | 0 | 78 | 0.00 | 1 | 0 |
| 200 | C0-200 | 1537 | 129 | 1.55 | 12 | 0 | 1539 | 2 | 129 | 1.55 | 12 | 0 |
| | C1-200 | 4216 | 85 | 3.53 | 35 | 1 | 4146 | 19 | 84 | 2.38 | 35 | 2 |
| | C2-200 | 4019 | 85 | 2.35 | 34 | 2 | 4018 | 16 | 85 | 3.53 | 33 | 3 |
| | C3-200 | 2029 | 113 | 0.00 | 13 | 0 | 2022 | 4 | 113 | 0.00 | 13 | 0 |
| | C4-200 | 2017 | 114 | 1.75 | 14 | 0 | 2039 | 4 | 114 | 1.75 | 13 | 0 |
| | C5-200 | 5161 | 80 | 1.25 | 82 | 9 | 5010 | 29 | 81 | 2.47 | 48 | 35 |
| | C6-200 | 4294 | 87 | 2.30 | 28 | 1 | 4315 | 18 | 87 | 2.30 | 25 | 3 |
| | C7-200 | 5516 | 76 | 2.63 | 81 | 67 | 5563 | 38 | 77 | 3.90 | 63 | 12 |
| | C8-200 | 5925 | 79 | 2.53 | 245 | 120 | 5986 | 29 | 79 | 2.53 | 209 | 120 |
| | C9-200 | 3142 | 99 | 1.01 | 16 | 2 | 3165 | 8 | 99 | 1.01 | 17 | 1 |
| | C10-200 | 4118 | 87 | 2.30 | 38 | 5 | 4150 | 10 | 87 | 2.30 | 30 | 3 |
| 300 | C0-300 | 9312 | 119 | 1.68 | 449 | 120 | 9366 | 58 | 119 | 1.68 | 235 | 120 |
| | C1-300 | 9928 | 119 | 2.52 | 376 | 120 | 9795 | 44 | 120 | 4.17 | 171 | 114 |
| | C2-300 | 4879 | 150 | 1.33 | 63 | 5 | 4894 | 3 | 150 | 1.33 | 62 | 6 |
| | C3-300 | 9465 | 114 | 1.75 | 286 | 62 | 9530 | 43 | 116 | 3.45 | 264 | 120 |
| | C4-300 | 7002 | 134 | 1.49 | 192 | 8 | 7021 | 24 | 134 | 1.49 | 113 | 14 |
| | C5-300 | 7920 | 125 | 0.80 | 296 | 120 | 7832 | 30 | 125 | 1.60 | 142 | 19 |
| | C6-300 | 9495 | 119 | 2.52 | 433 | 28 | 9531 | 70 | 119 | 3.36 | 157 | 14 |
| | C7-300 | 4311 | 158 | 1.27 | 63 | 4 | 4298 | 3 | 158 | 1.27 | 55 | 1 |
| | C8-300 | 9726 | 116 | 3.45 | 243 | 120 | 9756 | 31 | 115 | 2.61 | 219 | 120 |
| | C9-300 | 9796 | 115 | 3.48 | 401 | 120 | 9835 | 55 | 115 | 3.48 | 304 | 120 |
| | C10-300 | 3884 | 156 | 0.64 | 50 | 2 | 3915 | 8 | 156 | 1.28 | 49 | 1 |
| | C11-300 | 9354 | 119 | 4.20 | 303 | 120 | 9685 | 51 | 118 | 4.24 | 196 | 26 |
| | C12-300 | 4207 | 153 | 0.65 | 59 | 1 | 4285 | 6 | 153 | 0.65 | 53 | 1 |
| 400 | C0-400 | 10289 | 181 | 1.66 | 356 | 120 | 10355 | 27 | 182 | 2.20 | 318 | 120 |
| | C1-400 | 12853 | 152 | 5.92 | 417 | 120 | 12965 | 42 | 151 | 5.30 | 419 | 120 |
| | C2-400 | 9989 | 173 | 1.16 | 185 | 42 | 10112 | 19 | 173 | 1.16 | 203 | 21 |
| | C3-400 | 12615 | 155 | 11.61 | 486 | 120 | 12714 | 44 | 157 | 4.46 | 472 | 120 |
| | C4-400 | 8872 | 182 | 1.10 | 153 | 18 | 8948 | 22 | 183 | 2.19 | 146 | 26 |
| | C5-400 | 13162 | 154 | 3.25 | 415 | 120 | 13248 | 45 | 153 | 2.61 | 443 | 120 |
| | C6-400 | 8971 | 173 | 1.16 | 155 | 70 | 9130 | 28 | 173 | 1.73 | 159 | 25 |
| | C7-400 | 14322 | 150 | 6.67 | 504 | 120 | 14412 | 64 | 152 | 6.58 | 461 | 120 |

# APPENDIX B

# Appendix for Chapter 5

## B.1 The Pricing Subproblem $\text{PSP}_{\text{CTSPAV}}$

**Construction of graphs for $\text{PSP}_{\text{CTSPAV}}$** Let $v_t^i$ denote a virtual sink node for graph $\mathcal{G}_i^+$ or $\mathcal{G}_i^-$. Since a mini route covers only inbound trips or outbound trips, $\mathcal{G}_i^+ = (\mathcal{N}_i^+, \mathcal{A}_i^+)$ (resp. $\mathcal{G}_i^- = (\mathcal{N}_i^-, \mathcal{A}_i^-)$) contains, in addition to $v_t^i$, only the nodes for inbound trips (resp. outbound trips), i.e., $\mathcal{N}_i^+ = \mathcal{P}^+ \cup \mathcal{D}^+ \cup \{v_t^i\}$ (resp. $\mathcal{N}_i^- = \mathcal{P}^- \cup \mathcal{D}^- \cup \{v_t^i\}$). The set $\mathcal{A}_i^+$ (resp. $\mathcal{A}_i^-$) then represents all feasible edges for $\mathcal{G}_i^+$ (resp. $\mathcal{G}_i^-$), i.e., location pairs from $\mathcal{N}_i^+$ (resp. $\mathcal{N}_i^-$) that satisfy a priori route-feasibility constraints. Without loss of generality, the following elaborates further on how $\mathcal{G}_i^+$ is constructed.

Construction of $\mathcal{G}_i^+ = (\mathcal{N}_i^+, \mathcal{A}_i^+)$ begins with the introduction of the set of nodes $\mathcal{N}_i^+ = \mathcal{P}^+ \cup \mathcal{D}^+ \cup \{v_t^i\}$ and a set of fully-connected edges $\mathcal{A}_i^+ = \{(u, v) : u, v \in \mathcal{N}_i^+, u \neq v\}$. A ride-duration limit $L_u$ is associated with each node $u \in \mathcal{P}^+$, a time window $[a_u, b_u]$ and a service duration $\zeta_u$ are associated with each node $u \in \mathcal{P}^+ \cup \mathcal{D}^+$, and a travel time $\tau_{(u,v)}$ and a reduced cost $\bar{c}_{(u,v)}$ are associated with each edge $(u, v) \in \mathcal{A}_i^+$. As the goal is to find a feasible mini route from $i$ to $v_t^i$ with minimum reduced cost, $\bar{c}_{(u,v)}$ is defined as follows so that the total cost of any path from $i$ to $v_t^i$ is equivalent to that defined in (5.16).

$$\bar{c}_{(u,v)} = \begin{cases} -\pi_u - \mu_{(u,v)} & \forall u \in \mathcal{P}^+, \forall v \in \mathcal{P}^+ \cup \mathcal{D}^+ \\ -\mu_{(u,v)} & \forall u \in \mathcal{D}^+, \forall v \in \mathcal{P}^+ \cup \mathcal{D}^+ \\ 0 & \forall (u, v) \in \delta^-(v_t^i) \end{cases} \tag{B.1}$$

Edges from $\mathcal{A}_i^+$ that cannot belong to any feasible mini route are then identified by pre-processing time-window, pairing, precedence, and ride-duration limit constraints. Prior to this pre-processing step, knowledge of $i$ being the source of the path sought from $\mathcal{G}_i^+$ allows the time windows of all nodes $u \in \mathcal{P}^+ \cup \mathcal{D}^+$ to be tightened, by sequentially increasing their lower bounds using the following rules proposed by Dumas et al. (1991):

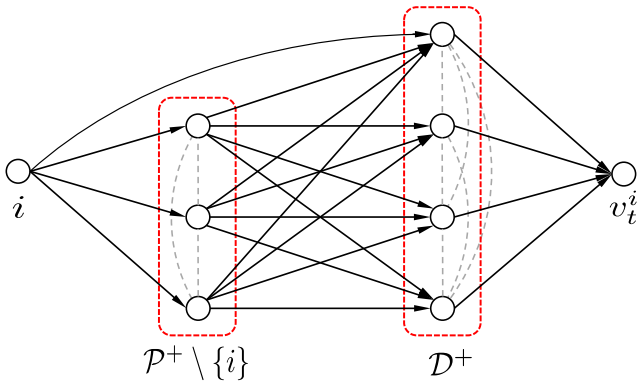



Figure B.1: Graph $\mathcal{G}_i^+$ (Each Dotted Line Represents a Pair of Bidirectional Edges).

- $a_u = \max\{a_u, a_i + \zeta_i + \tau_{(i,u)}\}, \forall u \in \mathcal{P}^+ \setminus \{i\}$
- $a_{n+u} = \max\{a_{n+u}, a_u + \zeta_u + \tau_{(u,n+u)}\}, \forall u \in \mathcal{P}^+ \setminus \{i\}$

The following sets of infeasible edges are then identified and consequently removed from $\mathcal{A}_i^+$:

(a) Direct trips to source $i$ and from sink $v_t^i$: $\delta^-(i) \cup \delta^+(v_t^i)$

(b) Precedence of pickup and drop-off nodes:
- $\{(i, v) : v \in \mathcal{D}^+ \setminus \{n + i\}\}$
- $\{(u, v_t^i) : u \in \mathcal{P}^+\}$
- $\{(u, v) : u \in \mathcal{D}^+ \wedge v \in \mathcal{P}^+\}$

(c) Time windows along each edge: $\{(u, v) : (u, v) \in \mathcal{A}_i^+ \setminus \delta^-(v_t^i) \wedge a_u + \zeta_u + \tau_{(u,v)} > b_v\}$

(d) Ride-duration limit of each commuter: $\{(u, v), (v, n+u) : u \in \mathcal{P}^+ \wedge v \in \mathcal{P}^+ \cup \mathcal{D}^+ \wedge u \neq v \wedge \tau_{(u,v)} + \zeta_v + \tau_{(v,n+u)} > L_u\}$

(e) Time windows and ride-duration limits of pairs of trips:
- $\{(u, n + v) : u, v \in \mathcal{P}^+ \wedge u \neq v \wedge \neg feasible(v \to u \to n + v \to n + u)\}$
- $\{(n + u, v) : u, v \in \mathcal{P}^+ \wedge u \neq v \wedge \neg feasible(u \to n + u \to v \to n + v)\}$
- $\{(u, v) : u, v \in \mathcal{P}^+ \wedge u \neq v \wedge \neg feasible(u \to v \to n + u \to n + v) \wedge \neg feasible(u \to v \to n + v \to n + u)\}$
- $\{(n + u, n + v) : u, v \in \mathcal{P}^+ \wedge u \neq v \wedge \neg feasible(u \to v \to n + u \to n + v) \wedge \neg feasible(v \to u \to n + u \to n + v)\}$

An example of graph $\mathcal{G}_i^+$ that results from the removal of the infeasible edges is shown in Figure B.1.

**Search for path with minimum reduced cost** The final step in $\text{PSP}_{\text{CTSPAV}}$ is to find a path from $i$ to $v_t^i$ from each graph $\mathcal{G}_i^+ (i \in \mathcal{P}^+)$ and $\mathcal{G}_i^- (i \in \mathcal{P}^-)$. On top of having the minimum cost, the path must also represent a *feasible* mini route. Notice that, as shown in Figure B.1, construction of the graphs eliminates all edges $\{(u, v) : u \in \mathcal{D} \wedge v \in \mathcal{P}\}$.

Therefore, any path from $i$ to $v_t^i$ is guaranteed to begin with a pickup phase that only visits nodes in $\mathcal{P}$ and end with a drop-off phase that only visits nodes in $\mathcal{D}$. In other words, the graphs ensure the precedence feasibility constraint, which requires all pickup nodes of a mini route to precede all of its drop-off nodes, is satisfied by construction. All that remains to guarantee the feasibility of any path found is to ensure that the path:

1. Visits each node within its specified time window,
2. Visits the corresponding drop-off nodes of every pickup node visited,
3. Satisfies the ride-duration limit of each rider served,
4. Respects the vehicle capacity, and
5. Visits each node at most once (i.e., the path is elementary).

The problem of finding a feasible, least-cost path from $i$ to $v_t^i$ is essentially an ESPPRC, whereby the remaining route-feasibility constraints can be modeled with resource constraints. In fact, this ESPPRC is identical to that in the column-generation pricing problem of the DARP considered in Gschwind and Irnich (2015). They proposed a dynamic-programming, label-setting algorithm that uses strong label-dominance rules to find the optimal solution to the problem.

While the algorithm can also be used to effectively solve this work's ESPPRC, it has a pre-requisite; it requires that the edge reduced costs $\bar{c}_{(u,v)}$ satisfy the delivery triangle inequality (DTI). The DTI, introduced by Ropke and Cordeau (2009), requires that the reduced costs fulfill $\bar{c}_{(u,v)} \leq \bar{c}_{(u,w)} + \bar{c}_{(w,v)}$ for all edges $(u,v),(u,w),(w,v) \in \{\mathcal{A}_i^+ : i \in \mathcal{P}^+\} \cup \{\mathcal{A}_i^- : i \in \mathcal{P}^-\}$ and $w \in \mathcal{D}$. Unfortunately, the reduced costs in this problem do not satisfy the DTI. A cost-matrix transformation, also proposed by Ropke and Cordeau (2009), is therefore first applied to the reduced costs to transform them into an equivalent set of costs that does satisfy the DTI, after which the label-setting algorithm of Gschwind and Irnich (2015) is applied on each graph $\mathcal{G}_i^+ (i \in \mathcal{P}^+)$ and $\mathcal{G}_i^- (i \in \mathcal{P}^-)$ to find the feasible, least-cost path from $i$ to $v_t^i$. Finally, note that enforcement of pairing and precedence resource constraints of the label-setting algorithm on the origin and destination node pairs of the graphs is sufficient to ensure elementarity of the paths produced, as the graphs lack edges $\{(u,v) : u \in \mathcal{D} \wedge v \in \mathcal{P}\}$ which are necessary to produce cycles in the presence of the pairing and precedence resource constraints. Therefore, additional resource constraints dedicated to specifically ensure elementarity of the paths are not necessary for $\text{PSP}_{\text{CTSPAV}}$.

## B.2 Optimality Gaps and Computation Times

The optimality gaps and computation times of all experiments for the CTSPAV procedure are summarized in Tables B.1 and B.2. In Table B.1, which summarizes the results of prob-

lem instances with the lexicographic objective, the first two columns specify the location of the clusters and the configuration of their depots. The following two columns list average values of the absolute gap for the vehicle count. The absolute gap is calculated by taking the vehicle count results of the MIP and subtracting from it its lower bound. The first gap uses the results from $\text{RMP}_{\text{CTSPAV}}$. Letting $Y_e^*$ be the value of $Y_e$ from $\text{RMP}_{\text{CTSPAV}}$ at convergence, the primal lower bound to the vehicle count is given by $\lceil \sum_{e \in \delta(v_s)} Y_e^* \rceil$. The second gap uses the primary objective results of $\text{RMP}_{\text{DARP}}$, particularly $\lceil z'_{\text{LB}} \rceil$ as the vehicle-count lower bound. It was found to consistently provide a stronger lower bound than $\lceil \sum_{e \in \delta(v_s)} Y_e^* \rceil$ from $\text{RMP}_{\text{CTSPAV}}$, and therefore it is included here to provide an additional perspective. The next column shows the average optimality gap, which is given by $(z_{\text{MIP}} - z^*)/z_{\text{MIP}}$ for each instance, where $z_{\text{MIP}}$ denotes the MIP's final objective value, and $z^*$, the objective value of $\text{RMP}_{\text{CTSPAV}}$ at convergence, provides a primal lower bound to $z_{\text{MIP}}$. However, for a few problem instances representing clusters outside the city limits, the column-generation phase did not converge within the time limit. For these instances, the dual lower bound $z_{\text{LB}}$ is used in place of $z^*$ when calculating the optimality gap, and the vehicle-count lower bound from the CTSPAV procedure is obtained by considering only the fixed cost contributions to $z_{\text{LB}}$. All uncertainties are represented by the standard error of the mean.

While the average optimality gap for the lexicographic objective is relatively high at approximately 70% (resp. 76%) for clusters inside the city (resp. outside the city), the vehicle-count gap, which depicts the absolute gap of the primary objective, paints a different picture, averaging at 2.5 and 7.0 vehicles for clusters inside and outside the city limits respectively. Moreover, when the lower bounds produced by the DARP procedure are used, the vehicle-count gap averages at even lower values of 1.1 and 4.2 vehicles for clusters inside and outside the city limits respectively. These lower values highlight a key strength of the DARP procedure: *It consistently produces stronger lower bounds for the primary objective*. When these stronger lower bounds are used, some problem instances produced an absolute vehicle-count gap of zero, indicating that their primary objective results are optimal.

The next column shows the average number of mini routes, i.e. columns, generated in the column-generation phase, while the following two columns summarize the time spent in this phase. The latter of the two summarizes the fraction of problem instances in which the column-generation phase met or exceeded the 1-hour time limit (i.e., the fraction of instances whereby the column-generation phase did not converge within the time limit), while the former shows the average wall time spent on column generation *for problem instances that did not exceed the time limit*. It can be seen that on average, the time spent

Table B.1: Optimality Gaps and Computation Times of the CTSPAV Procedure with the Lexicographic Objective

| Cluster location | Depot config. | Average vehicle count gap | Average vehicle count gap (DARP LB) | Average optimality gap (%) | Average mini route count | Average colgen time (s) | % ≥ colgen time limit | % ≥ MIP time limit |
|---|---|---|---|---|---|---|---|---|
| Inside city | Central | $2.55 \pm 0.14$ | $1.09 \pm 0.15$ | $70.5 \pm 1.4$ | $5394 \pm 741$ | $86 \pm 28$ | 0.0 | 100.0 |
| | Local | $2.50 \pm 0.16$ | $1.09 \pm 0.16$ | $69.8 \pm 1.7$ | $5699 \pm 893$ | $161 \pm 84$ | 0.0 | 100.0 |
| Outside city | Central | $7.07 \pm 0.45$ | $4.19 \pm 0.28$ | $75.5 \pm 2.1$ | $19949 \pm 1021$ | $926 \pm 141$ | 32.4 | 100.0 |
| | Local | $7.03 \pm 0.47$ | $4.12 \pm 0.30$ | $75.5 \pm 2.2$ | $19631 \pm 974$ | $922 \pm 133$ | 36.8 | 100.0 |

Table B.2: Optimality Gaps and Computation Times of the CTSPAV Procedure with the Distance-Minimization Objective

| Cluster location | Depot config. | Average optimality gap (%) | Average mini route count | Average colgen time (s) | % ≥ colgen time limit | % ≥ MIP time limit |
|---|---|---|---|---|---|---|
| Inside city | Central | $1.75 \pm 0.17$ | $5396 \pm 752$ | $73 \pm 22$ | 0.0 | 36.4 |
| | Local | $1.35 \pm 0.17$ | $5453 \pm 803$ | $105 \pm 37$ | 0.0 | 13.6 |
| Outside city | Central | $3.05 \pm 0.48$ | $20533 \pm 1147$ | $960 \pm 112$ | 8.8 | 82.4 |
| | Local | $4.75 \pm 1.04$ | $21114 \pm 1140$ | $1126 \pm 129$ | 19.1 | 73.5 |

on column generation is significantly higher for clusters outside the city limits. On top of that, more than 30% of the instances did not achieve convergence within the time limit. This can be attributed to the significantly larger number of columns generated from these clusters. Finally, the last column shows the fraction of problem instances in which the MIP-solving phase met or exceeded its 1-hour time budget. For the lexicographic objective, the time limit was exceeded by all problem instances.

Table B.2 summarizes the results of problem instances with the distance-minimization objective. It displays the same information as Table B.1, except that it does not list the average vehicle-count gaps. This is due to the lower bound of the vehicle count not being available from the objective. It can be seen that the optimality gaps for the distance-minimization objective are excellent, being less than 2% on average for cases inside the city and less than 5% on average for cases outside. Similar to the instances with the lexicographic objective, the average number of columns generated from clusters outside the city limits is significantly larger than that from clusters inside, which consequently results in the significantly longer average time spent on the column-generation phase. Finally, the percentage of problem instances that exceed the time limits of the column-generation and

the MIP phases is fewer than that for the lexicographic objective in every case, and this can be attributed to the relatively stronger primal lower bound provided by the LP relaxation of $\text{MP}_{\text{CTSPAV}}$ when the distance-minimization objective is utilized.

Table B.3 summarizes the average optimality gaps and computations times of the same set of problem instances for the DARP procedure with the lexicographic objective. Once again, all uncertainties are represented by the standard error of the mean. Its first two columns specify the cluster location and the depot configuration of the instances. The next lists the average number of columns generated, followed by two columns which show the average absolute gap and the average optimality gap of *the primary objective value*. Letting $z'_{\text{MIP}}$ denote the final primary objective value of the MIP and recalling that $z'_{\text{LB}}$ denotes its dual lower bound, the absolute gap of an instance given by $z'_{\text{MIP}} - \lceil z'_{\text{LB}} \rceil$ while its optimality gap is given by $(z'_{\text{MIP}} - \lceil z'_{\text{LB}} \rceil)/z'_{\text{MIP}}$. The average optimality gaps are relatively high at approximately 48%. However, the absolute gaps reveal a different story, whereby they average at only 2.6 and 4.8 vehicles inside and outside the city limits respectively. These values, however, are higher than those for the CTSPAV procedure which utilize the same lower bounds ($\lceil z'_{\text{LB}} \rceil$).

The next column specifies the fraction of problem instances whereby the column-generation phase for the primary objective did not converge within the time limit, followed by one which specifies the fraction of instances whereby the column-generation phase for both objectives did not converge. When only the primary objective is considered, the column generation for more than two-thirds of the instances inside the city and almost all instances outside the city did not converge. On top of that, out of the 180 problem instances in total, only two had their column-generation phases for both objectives converge within the allocated time limit, even after more time has been allocated for this phase (recall that 1.5 hours is allocated for this phase of the DARP procedure as opposed to only 1 hour for the CTSPAV procedure). This highlights the harder nature of $\text{PSP}_{\text{DARP}}$ which searches for the longer AV routes (as opposed to $\text{PSP}_{\text{CTSPAV}}$ which only searches for mini routes). Nevertheless, even though its column-generation phase did not converge in most instances, *the DARP procedure was still able to consistently produce stronger lower bounds for the primary objective*.

The next column shows the fraction of problem instances in which the MIP for the primary objective exceeded the time limit, while the last shows the fraction of instances in which the MIP for both objectives exceeded the time limit. Both columns show that the MIP can be solved within the time limit for a majority of the problem instances, which is in stark contrast to the MIP of the CTSPAV procedure, whereby its longer time limit is exceeded by all problem instances. This difference can be attributed to the relatively easier

Table B.3: Optimality Gaps and Computation Times of the DARP Procedure with the Lexicographic Objective

| Cluster location | Depot config. | Average column count | Average primary absolute gap | Average primary optimality gap (%) | % ≥ primary colgen time limit | % ≥ colgen time limit | % ≥ primary MIP time limit | % ≥ MIP time limit |
|---|---|---|---|---|---|---|---|---|
| Inside city | Central | $34107 \pm 5602$ | $2.68 \pm 0.14$ | $47.9 \pm 2.1$ | 68.2 | 100.0 | 18.2 | 36.4 |
| | Local | $33911 \pm 5554$ | $2.55 \pm 0.16$ | $49.0 \pm 2.7$ | 68.2 | 100.0 | 9.1 | 27.3 |
| Outside city | Central | $6801 \pm 907$ | $4.81 \pm 0.23$ | $48.2 \pm 1.5$ | 95.6 | 98.5 | 14.7 | 25.0 |
| | Local | $6698 \pm 870$ | $4.79 \pm 0.25$ | $47.6 \pm 1.6$ | 95.6 | 98.5 | 14.7 | 19.1 |

$\text{MP}_{\text{DARP}}$ which just solves a set-covering problem (instead of the $\text{MP}_{\text{CTSPAV}}$ which solves a route-scheduling problem) and the stronger primal lower bound provided by its linear relaxation.

Table B.4 summarizes the average optimality gaps and computations times for the DARP procedure with the distance-minimization objective. Similar to Table B.3, the first three columns specify the cluster locations, the depot configurations, and the average number of columns generated. The following two show the average optimality gaps of the final MIP results. The first uses the dual lower bound, $\lceil z_{\text{LB}} \rceil$, in the gap calculation, whereas the second uses the primal lower bound produced by the CTSPAV procedure, $z^*$. For the distance-minimization objective, the CTSPAV procedure consistently produces stronger lower bounds, therefore the optimality gaps calculated using its lower bound are also smaller. The dual lower bound $\lceil z_{\text{LB}} \rceil$ is weaker in this case because the column-generation phase could not be completed within the time limit for almost all problem instances. This is further highlighted by the next column which shows the fraction of problems in which the column-generation phase could not be completed. Out of the 180 problem instances in total, only four managed to achieve convergence. For these four instances, the strength of $\lceil z_{\text{LB}} \rceil$ is comparable to $z^*$ from the CTSPAV procedure. Finally, the last column shows the fraction of problem instances in which the MIP exceeded its time limit. A larger fraction of instances inside the city exceeded the time limit due to their relatively larger column counts.

Table B.4: Optimality Gaps and Computation Times of the DARP Procedure with the Distance Minimization Objective

| Cluster location | Depot config. | Average column count | Average optimality gap (%) | Average optimality gap (%) (CTSPAV LB) | % $\geq$ colgen time limit | % $\geq$ MIP time limit |
|---|---|---|---|---|---|---|
| Inside city | Central | $34286 \pm 5323$ | $98.0 \pm 2.0$ | $28.5 \pm 1.7$ | 100.0 | 77.3 |
| | Local | $30712 \pm 4640$ | $95.2 \pm 3.8$ | $36.4 \pm 3.1$ | 100.0 | 77.3 |
| Outside city | Central | $6154 \pm 902$ | $94.8 \pm 2.6$ | $28.4 \pm 1.6$ | 96.7 | 36.7 |
| | Local | $5816 \pm 838$ | $95.5 \pm 2.6$ | $44.5 \pm 1.8$ | 96.9 | 37.5 |

# APPENDIX C

# Appendix for Chapter 6

## C.1 Filtering of Graph $\mathcal{G}$

The following sets of infeasible edges, identified using rules proposed by Dumas et al. (1991) and Cordeau (2006) which pre-process time-window, pairing, precedence, and ride-duration limit constraints on $\mathcal{A}$, are eliminated to produce a more compact representation of graph $\mathcal{G}$:

(a) Direct trips to and from the depot:
- $\{(v_s, v_t), (v_t, v_s)\}$
- $\{(i, v_s), (i, v_t), (v_t, i) : i \in \mathcal{P}\}$
- $\{(v_s, i), (i, v_s), (v_t, i) : i \in \mathcal{D}\}$

(b) Precedence of pickup and drop-off nodes of inbound and outbound trips of each commuter (constraints (6.1)): $\{(i, 2n+i), (i, 3n+i), (n+i, i), (n+i, 3n+i), (2n+i, i), (2n+i, n+i), (3n+i, i), (3n+i, n+i), (3n+i, 2n+i) : i \in \mathcal{P}^+\}$

(c) Precedence of pickup and drop-off nodes of inbound and outbound mini routes:
- $\{(i, j) : i \in \mathcal{P}^+ \wedge j \in \mathcal{P}^- \cup \mathcal{D}^-\}$
- $\{(i, j) : i \in \mathcal{D}^+ \wedge j \in \mathcal{D}^-\}$
- $\{(i, j) : i \in \mathcal{P}^- \wedge j \in \mathcal{P}^+ \cup \mathcal{D}^+\}$
- $\{(i, j) : i \in \mathcal{D}^- \wedge j \in \mathcal{D}^+\}$

(d) Time windows along each edge: $\{(i, j) : (i, j) \in \mathcal{A} \setminus \{\delta^+(v_s) \cup \delta^-(v_t)\} \wedge a_i + \zeta_i + \tau_{(i,j)} > b_j\}$

(e) Ride-duration limit of each commuter: $\{(i, j), (j, n+i) : i \in \mathcal{P} \wedge j \in \mathcal{P} \cup \mathcal{D} \wedge i \neq j \wedge \tau_{(i,j)} + \zeta_j + \tau_{(j,n+i)} > L_i\}$

(f) Time windows and ride-duration limits of pairs of trips:
- $\{(i, n+j) : i, j \in \mathcal{P} \wedge i \neq j \wedge \neg feasible(j \rightarrow i \rightarrow n+j \rightarrow n+i)\}$
- $\{(n+i, j) : i, j \in \mathcal{P} \wedge i \neq j \wedge \neg feasible(i \rightarrow n+i \rightarrow j \rightarrow n+j)\}$
- $\{(i, j) : i, j \in \mathcal{P} \wedge i \neq j \wedge \neg feasible(i \rightarrow j \rightarrow n+i \rightarrow n+j) \wedge \neg feasible(i \rightarrow$

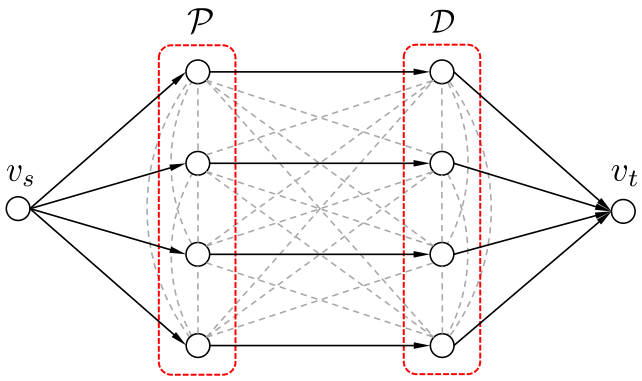



Figure C.1: Graph $\mathcal{G}$ (Each Dotted Line Represents a Pair of Bidirectional Edges).

$j \rightarrow n+j \rightarrow n+i)\}$

- $\{(n+i, n+j) : i, j \in \mathcal{P} \wedge i \neq j \wedge \neg feasible(i \rightarrow j \rightarrow n+i \rightarrow n+j) \wedge \neg feasible(j \rightarrow i \rightarrow n+i \rightarrow n+j)\}$

Note that the sets of edges in (f) utilize the $feasible$ function described in Section 1.1.3 to determine if a partial route satisfies the time-window and ride-duration limit constraints. For instance, the first condition indicates that edge $(i, n+j)$ is infeasible if the partial route $j \rightarrow i \rightarrow n+j \rightarrow n+i$ is infeasible. Figure C.1 illustrates an example of graph $\mathcal{G}$ resulting from this pre-processing step.

## C.2 Computational Results

Table C.1 summarizes the results of CTSPAV$_{\text{Hybrid}}$ for every large problem instance. Its first column shows the name of every instance. The next three columns display properties that characterize the size of each instance. They list the node count of graph $\mathcal{G}$, $|\mathcal{N}|$, the edge count of the graph (after the pre-processing step), $|\mathcal{A}|$, and finally the number of mini routes generated by the MREA, $|\Omega|$, for every instance. The next column shows the wall time spent to the enumerate the mini routes. The remaining columns summarize the results of CTSPAV$_{\text{Hybrid}}$. The first two show the vehicle count and total travel distance from its best incumbent solution. The next two display the absolute gap for the vehicle count and the optimality gap for the objective value of the best incumbent solution. The following column shows the number of tree nodes explored in the solution process. The last two columns display the (total) wall time spent to solve the MIP and that spent to close the vehicle count gap. For the very last column, values are only listed for instances whereby the vehicle count gap could be closed within the 2-hour time limit. It is left blank otherwise. Tables C.3 and C.5 provide the same set of information for CTSPAV$_{\text{Hybrid}}$ for

every medium and tight problem instance respectively. On the other hand, Tables C.2, C.4, and C.6 show the results of $\text{CTSPAV}_{\text{SEC}}$ and $\text{CTSPAV}_{\text{Base}}$ for all large, medium, and tight problem instances respectively.

Table C.1: Results of $\text{CTSPAV}_{\text{Hybrid}}$ for the Large Problem Instances

| Instance name | Node count | Edge count | Mini route count | Route enumeration time (s) | Vehicle count | Total distance (m) | Vehicle count gap | Optimality gap (%) | Nodes explored | Wall time (s) | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | MIP | Optimal count |
| L0 | 402 | 23983 | 3730 | 22 | 3 | 642049 | 0 | 0.0 | 156016 | 5360 | 1284 |
| L1 | 402 | 22621 | 1093 | 21 | 3 | 463065 | 1 | 33.3 | 524584 | 7200 | - |
| L2 | 402 | 26781 | 51175 | 24 | 4 | 817348 | 2 | 49.9 | 6424 | 7200 | - |
| L3 | 402 | 26496 | 63597 | 24 | 4 | 841180 | 2 | 49.9 | 7430 | 7202 | - |
| L4 | 402 | 25309 | 49147 | 23 | 4 | 813018 | 1 | 24.9 | 11734 | 7201 | - |
| L5 | 402 | 22425 | 1605 | 20 | 3 | 512675 | 1 | 33.3 | 189596 | 7200 | - |
| L6 | 402 | 26420 | 20060 | 23 | 4 | 955285 | 2 | 49.9 | 7935 | 7201 | - |
| L7 | 402 | 24699 | 21403 | 23 | 4 | 888490 | 1 | 24.9 | 22067 | 7201 | - |
| L8 | 402 | 25710 | 14818 | 23 | 4 | 844674 | 1 | 24.9 | 23822 | 7200 | - |
| L9 | 402 | 27315 | 191067 | 25 | 5 | 737361 | 3 | 59.9 | 1511 | 7200 | - |
| L10 | 402 | 24386 | 5807 | 25 | 3 | 555102 | 1 | 33.3 | 30016 | 7201 | - |
| L11 | 402 | 25639 | 18237 | 23 | 3 | 570036 | 1 | 33.3 | 13176 | 7201 | - |
| L12 | 402 | 23748 | 3631 | 21 | 3 | 581863 | 1 | 33.3 | 125059 | 7200 | - |
| L13 | 402 | 24581 | 6835 | 24 | 3 | 624843 | 1 | 33.3 | 23394 | 7202 | - |
| L14 | 402 | 26287 | 72200 | 23 | 4 | 949361 | 2 | 49.9 | 5138 | 7201 | - |
| L15 | 402 | 24898 | 114817 | 38 | 4 | 1108007 | 2 | 49.9 | 7258 | 7200 | - |
| L16 | 402 | 24203 | 9231 | 22 | 4 | 847394 | 1 | 24.9 | 75500 | 7200 | - |
| L17 | 402 | 23734 | 6404 | 22 | 4 | 863265 | 0 | 0.0 | 22485 | 7200 | 5883 |
| L18 | 402 | 24712 | 4417 | 33 | 4 | 914762 | 1 | 24.9 | 33188 | 7201 | - |
| L19 | 402 | 25513 | 35873 | 24 | 3 | 698599 | 1 | 33.3 | 11984 | 7201 | - |
| L20 | 402 | 25528 | 58833 | 23 | 3 | 779684 | 1 | 33.3 | 8639 | 7200 | - |
| L21 | 402 | 22832 | 4870 | 21 | 2 | 457911 | 0 | 0.0 | 166142 | 7200 | 2217 |

Table C.2: Results of $\text{CTSPAV}_{\text{SEC}}$ and $\text{CTSPAV}_{\text{Base}}$ for the Large Problem Instances

| Instance name | CTSPAV variant | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SEC | | | | | | | Base | | | | | | |
| | Vehicle count | Total distance (m) | Vehicle count gap | Optimality gap (%) | Nodes explored | Wall time (s) | | Vehicle count | Total distance (m) | Vehicle count gap | Optimality gap (%) | Nodes explored | Wall time (s) | |
| | | | | | | MIP | Optimal count | | | | | | MIP | Optimal count |
| L0 | 3 | 646884 | 1 | 33.3 | 43103 | 7200 | - | 3 | 652906 | 2 | 66.5 | 24638 | 7201 | - |
| L1 | 3 | 463065 | 1 | 33.3 | 135613 | 7228 | - | 3 | 463065 | 2 | 66.6 | 408157 | 7202 | - |
| L2 | 4 | 821989 | 2 | 49.9 | 6369 | 7218 | - | 4 | 824321 | 3 | 74.8 | 5229 | 7201 | - |
| L3 | 4 | 849844 | 2 | 49.9 | 4713 | 7215 | - | 4 | 843208 | 3 | 74.8 | 5291 | 7200 | - |
| L4 | 5 | 820800 | 3 | 59.9 | 10005 | 7202 | - | 5 | 831319 | 3 | 59.9 | 20952 | 7201 | - |
| L5 | 3 | 512838 | 1 | 33.3 | 73463 | 7202 | - | 3 | 512675 | 2 | 66.5 | 195089 | 7201 | - |
| L6 | 4 | 971911 | 2 | 49.9 | 9541 | 7207 | - | 4 | 967746 | 3 | 74.8 | 11540 | 7204 | - |
| L7 | 4 | 891808 | 2 | 49.9 | 7244 | 7206 | - | 4 | 893550 | 3 | 74.8 | 15275 | 7201 | - |
| L8 | 4 | 845333 | 2 | 49.9 | 8301 | 7201 | - | 4 | 845100 | 3 | 74.8 | 16814 | 7200 | - |
| L9 | 5 | 730915 | 3 | 59.9 | 2023 | 7200 | - | 5 | 720023 | 4 | 79.9 | 1906 | 7200 | - |
| L10 | 3 | 555102 | 1 | 33.3 | 21162 | 7200 | - | 3 | 555102 | 2 | 66.5 | 21223 | 7201 | - |
| L11 | 3 | 573246 | 1 | 33.3 | 3428 | 7203 | - | 3 | 574227 | 2 | 66.5 | 21195 | 7200 | - |
| L12 | 3 | 581863 | 1 | 33.3 | 34193 | 7200 | - | 3 | 581863 | 2 | 66.5 | 57588 | 7202 | - |
| L13 | 3 | 626100 | 1 | 33.3 | 15871 | 7221 | - | 3 | 625042 | 2 | 66.5 | 36251 | 7201 | - |
| L14 | 4 | 949659 | 2 | 49.9 | 5431 | 7213 | - | 4 | 932389 | 3 | 74.8 | 4986 | 7200 | - |
| L15 | 4 | 1108620 | 2 | 49.9 | 4435 | 7202 | - | 4 | 1116187 | 3 | 74.8 | 2732 | 7201 | - |
| L16 | 4 | 857161 | 2 | 49.9 | 12595 | 7203 | - | 4 | 846684 | 3 | 74.8 | 21489 | 7200 | - |
| L17 | 4 | 867674 | 2 | 49.9 | 21259 | 7201 | - | 4 | 865011 | 2 | 49.9 | 21691 | 7200 | - |
| L18 | 4 | 917395 | 2 | 49.9 | 18251 | 7201 | - | 4 | 914762 | 2 | 49.9 | 20825 | 7200 | - |
| L19 | 4 | 697540 | 2 | 49.9 | 4925 | 7298 | - | 4 | 706887 | 3 | 74.9 | 15757 | 7200 | - |
| L20 | 3 | 772418 | 1 | 33.3 | 6277 | 7318 | - | 3 | 778248 | 2 | 66.5 | 7573 | 7200 | - |
| L21 | 3 | 447435 | 2 | 66.6 | 1632 | 7259 | - | 2 | 458460 | 1 | 49.9 | 86453 | 7205 | - |

Table C.3: Results of $\text{CTSPAV}_{\text{Hybrid}}$ for the Medium Problem Instances

| Instance name | Node count | Edge count | Mini route count | Route enumeration time (s) | Vehicle count | Total distance (m) | Vehicle count gap | Optimality gap (%) | Nodes explored | Wall time (s) | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | MIP | Optimal count |
| M0 | 302 | 14024 | 3233 | 7 | 2 | 481141 | 0 | 0.0 | 109840 | 7200 | 445 |
| M1 | 262 | 11267 | 8986 | 6 | 3 | 605515 | 1 | 33.3 | 39142 | 7200 | - |
| M2 | 302 | 13973 | 31559 | 7 | 3 | 847030 | 0 | 0.0 | 27300 | 7200 | 4567 |
| M3 | 302 | 15253 | 30739 | 10 | 3 | 668490 | 1 | 33.3 | 18968 | 7201 | - |
| M4 | 302 | 14426 | 28359 | 9 | 3 | 535195 | 1 | 33.3 | 19036 | 7201 | - |
| M5 | 302 | 12739 | 503 | 6 | 2 | 333048 | 0 | 0.0 | 1348803 | 3409 | 340 |
| M6 | 302 | 15515 | 47521 | 8 | 3 | 657988 | 1 | 33.3 | 12023 | 7200 | - |
| M7 | 302 | 14485 | 3485 | 7 | 3 | 595519 | 1 | 33.3 | 123341 | 7200 | - |
| M8 | 302 | 15404 | 10828 | 8 | 3 | 689147 | 1 | 33.3 | 21890 | 7201 | - |
| M9 | 302 | 15882 | 55026 | 9 | 3 | 489997 | 1 | 33.3 | 14828 | 7201 | - |
| M10 | 302 | 14898 | 119198 | 10 | 3 | 719639 | 1 | 33.3 | 18473 | 7200 | - |
| M11 | 302 | 13800 | 5845 | 10 | 2 | 602968 | 0 | 0.0 | 205444 | 7200 | 1814 |
| M12 | 302 | 13542 | 1884 | 7 | 2 | 417175 | 0 | 0.0 | 61043 | 1007 | 122 |
| M13 | 302 | 14564 | 28922 | 9 | 3 | 652724 | 1 | 33.3 | 18510 | 7200 | - |
| M14 | 302 | 13902 | 3207 | 7 | 2 | 401064 | 0 | 0.0 | 51325 | 2406 | 270 |
| M15 | 302 | 14801 | 14693 | 7 | 3 | 627967 | 0 | 0.0 | 39332 | 7200 | 7030 |
| M16 | 254 | 10233 | 3968 | 4 | 3 | 599126 | 0 | 0.0 | 30465 | 2949 | 2787 |
| M17 | 302 | 13224 | 1380 | 7 | 2 | 490178 | 0 | 0.0 | 14669 | 134 | 73 |
| M18 | 290 | 11758 | 749 | 5 | 2 | 347259 | 0 | 0.0 | 30780 | 418 | 416 |
| M19 | 302 | 13043 | 3174 | 7 | 2 | 339073 | 0 | 0.0 | 278853 | 6566 | 6004 |
| M20 | 302 | 14184 | 4380 | 7 | 3 | 551547 | 1 | 33.3 | 81164 | 7200 | - |
| M21 | 258 | 10135 | 1696 | 6 | 3 | 620764 | 0 | 0.0 | 273752 | 4256 | 4116 |
| M22 | 302 | 14856 | 19435 | 8 | 3 | 683612 | 1 | 33.3 | 18247 | 7200 | - |
| M23 | 302 | 14230 | 12339 | 7 | 3 | 556522 | 1 | 33.3 | 31373 | 7200 | - |
| M24 | 302 | 14694 | 23970 | 7 | 3 | 588191 | 1 | 33.3 | 18586 | 7200 | - |
| M25 | 286 | 13139 | 19056 | 6 | 3 | 596412 | 1 | 33.3 | 24223 | 7201 | - |
| M26 | 302 | 13505 | 1547 | 11 | 3 | 445952 | 0 | 0.0 | 55454 | 1576 | 1311 |
| M27 | 262 | 10980 | 4981 | 4 | 3 | 712881 | 0 | 0.0 | 34804 | 1648 | 1422 |
| M28 | 302 | 13883 | 3565 | 11 | 2 | 394323 | 0 | 0.0 | 1737 | 183 | 104 |
| M29 | 302 | 15142 | 38021 | 10 | 3 | 729149 | 1 | 33.3 | 18677 | 7200 | - |

Table C.4: Results of $\text{CTSPAV}_{\text{SEC}}$ and $\text{CTSPAV}_{\text{Base}}$ for the Medium Problem Instances

| Instance name | CTSPAV variant | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SEC | | | | | | | Base | | | | | | |
| | Vehicle count | Total distance (m) | Vehicle count gap | Optimality gap (%) | Nodes explored | Wall time (s) | | Vehicle count | Total distance (m) | Vehicle count gap | Optimality gap (%) | Nodes explored | Wall time (s) | |
| | | | | | | MIP | Optimal count | | | | | | MIP | Optimal count |
| M0 | 2 | 480223 | 0 | 0.0 | 229608 | 7200 | 2756 | 2 | 480225 | 1 | 49.9 | 143747 | 7201 | - |
| M1 | 3 | 603771 | 1 | 33.3 | 21394 | 7203 | - | 3 | 604103 | 2 | 66.5 | 20833 | 7200 | - |
| M2 | 3 | 846579 | 1 | 33.2 | 16221 | 7207 | - | 3 | 846597 | 1 | 33.2 | 21540 | 7201 | - |
| M3 | 3 | 668248 | 1 | 33.3 | 15377 | 7205 | - | 3 | 682726 | 2 | 66.5 | 21125 | 7200 | - |
| M4 | 3 | 535334 | 1 | 33.3 | 7076 | 7298 | - | 3 | 535195 | 2 | 66.5 | 21423 | 7200 | - |
| M5 | 2 | 333048 | 0 | 0.0 | 14122 | 335 | 95 | 2 | 333366 | 1 | 49.9 | 1119738 | 7200 | - |
| M6 | 3 | 656983 | 1 | 33.3 | 6422 | 7201 | - | 4 | 655969 | 3 | 74.9 | 20905 | 7200 | - |
| M7 | 3 | 595519 | 1 | 33.3 | 45152 | 7204 | - | 3 | 595519 | 2 | 66.5 | 65095 | 7201 | - |
| M8 | 3 | 679167 | 1 | 33.3 | 21493 | 7201 | - | 3 | 687498 | 2 | 66.5 | 21259 | 7200 | - |
| M9 | 3 | 489461 | 1 | 33.3 | 5734 | 7238 | - | 3 | 497878 | 2 | 66.6 | 21032 | 7200 | - |
| M10 | 3 | 719788 | 1 | 33.3 | 8781 | 7215 | - | 3 | 722278 | 2 | 66.5 | 3697 | 7201 | - |
| M11 | 2 | 601111 | 0 | 0.0 | 35354 | 7202 | 1730 | 2 | 601041 | 1 | 49.8 | 29943 | 7200 | - |
| M12 | 2 | 417175 | 0 | 0.0 | 50401 | 1911 | 195 | 2 | 417185 | 1 | 49.9 | 251358 | 7200 | - |
| M13 | 3 | 655996 | 1 | 33.3 | 7966 | 7212 | - | 3 | 653183 | 2 | 66.5 | 21185 | 7202 | - |
| M14 | 2 | 401064 | 0 | 0.0 | 26183 | 5314 | 983 | 2 | 401064 | 1 | 49.9 | 92983 | 7200 | - |
| M15 | 4 | 622760 | 2 | 49.9 | 20584 | 7203 | - | 4 | 622717 | 2 | 49.9 | 23019 | 7200 | - |
| M16 | 3 | 599126 | 1 | 33.3 | 80256 | 7205 | - | 3 | 599442 | 2 | 66.5 | 32695 | 7200 | - |
| M17 | 2 | 490178 | 0 | 0.0 | 4120 | 141 | 58 | 2 | 490178 | 1 | 49.9 | 272323 | 7200 | - |
| M18 | 2 | 347259 | 0 | 0.0 | 436 | 156 | 151 | 2 | 347259 | 1 | 49.9 | 1064235 | 7201 | - |
| M19 | 2 | 339073 | 0 | 0.0 | 4695 | 1645 | 637 | 2 | 339073 | 1 | 49.9 | 192573 | 7200 | - |
| M20 | 3 | 551547 | 1 | 33.3 | 41920 | 7203 | - | 3 | 551547 | 2 | 66.5 | 39175 | 7200 | - |
| M21 | 3 | 620783 | 1 | 33.3 | 211984 | 7200 | - | 3 | 620764 | 2 | 66.5 | 319796 | 7200 | - |
| M22 | 3 | 685043 | 1 | 33.3 | 15662 | 7205 | - | 3 | 683300 | 1 | 33.3 | 24972 | 7200 | - |
| M23 | 3 | 556571 | 1 | 33.3 | 21292 | 7200 | - | 3 | 555996 | 2 | 66.5 | 20915 | 7200 | - |
| M24 | 3 | 588191 | 1 | 33.3 | 17174 | 7223 | - | 3 | 587860 | 2 | 66.5 | 21374 | 7200 | - |
| M25 | 3 | 597367 | 1 | 33.3 | 20807 | 7200 | - | 3 | 596653 | 2 | 66.5 | 21527 | 7201 | - |
| M26 | 3 | 445952 | 1 | 33.3 | 114827 | 7202 | - | 3 | 445952 | 2 | 66.6 | 189359 | 7200 | - |
| M27 | 3 | 712881 | 1 | 33.3 | 25439 | 7202 | - | 3 | 712881 | 2 | 66.5 | 21500 | 7200 | - |
| M28 | 2 | 394323 | 0 | 0.0 | 1830 | 431 | 241 | 2 | 394323 | 1 | 49.9 | 139970 | 7200 | - |
| M29 | 3 | 731148 | 1 | 33.3 | 10958 | 7204 | - | 3 | 729946 | 2 | 66.5 | 19975 | 7203 | - |

Table C.5: Results of $\text{CTSPAV}_{\text{Hybrid}}$ for the Tight Problem Instances

| Instance name | Node count | Edge count | Mini route count | Route enumeration time (s) | Vehicle count | Total distance (m) | Vehicle count gap | Optimality gap (%) | Nodes explored | Wall time (s) | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | MIP | Optimal count |
| S0 | 402 | 20870 | 374 | 19 | 5 | 961566 | 0 | 0.0 | 144186 | 544 | 129 |
| S1 | 402 | 20847 | 267 | 18 | 3 | 619257 | 0 | 0.0 | 19909 | 143 | 124 |
| S2 | 402 | 21424 | 971 | 20 | 5 | 1246019 | 0 | 0.0 | 27515 | 459 | 333 |
| S3 | 402 | 21472 | 1268 | 21 | 5 | 1192722 | 0 | 0.0 | 19049 | 830 | 721 |
| S4 | 402 | 21352 | 1204 | 20 | 5 | 1187914 | 0 | 0.0 | 957524 | 5084 | 238 |
| S5 | 402 | 20918 | 304 | 17 | 3 | 676142 | 0 | 0.0 | 1887 | 28 | 24 |
| S6 | 402 | 21050 | 707 | 20 | 6 | 1503404 | 0 | 0.0 | 14494 | 224 | 187 |
| S7 | 402 | 21022 | 687 | 20 | 5 | 1345009 | 0 | 0.0 | 121198 | 1524 | 1180 |
| S8 | 402 | 20896 | 581 | 31 | 5 | 1310231 | 0 | 0.0 | 2705 | 37 | 32 |
| S9 | 402 | 21876 | 1666 | 30 | 6 | 1094536 | 0 | 0.0 | 14475 | 384 | 262 |
| S10 | 402 | 21044 | 430 | 29 | 4 | 805606 | 0 | 0.0 | 17905 | 228 | 228 |
| S11 | 402 | 21614 | 835 | 29 | 4 | 819652 | 0 | 0.0 | 11194 | 211 | 188 |
| S12 | 402 | 20946 | 393 | 32 | 4 | 837723 | 0 | 0.0 | 448878 | 1504 | 86 |
| S13 | 402 | 21137 | 504 | 20 | 4 | 914708 | 0 | 0.0 | 136179 | 1149 | 667 |
| S14 | 402 | 21438 | 1056 | 32 | 5 | 1450697 | 0 | 0.0 | 10064 | 71 | 17 |
| S15 | 402 | 21156 | 2825 | 31 | 5 | 1613836 | 0 | 0.0 | 2646 | 20 | 8 |
| S16 | 402 | 21005 | 528 | 32 | 5 | 1220586 | 0 | 0.0 | 8396 | 147 | 136 |
| S17 | 402 | 20844 | 499 | 30 | 5 | 1252397 | 0 | 0.0 | 9523 | 68 | 34 |
| S18 | 402 | 20713 | 392 | 31 | 6 | 1452716 | 0 | 0.0 | 18044 | 201 | 200 |
| S19 | 402 | 21377 | 1267 | 31 | 4 | 1030225 | 0 | 0.0 | 513121 | 4069 | 1218 |
| S20 | 402 | 21542 | 1541 | 33 | 4 | 1144849 | 0 | 0.0 | 8369 | 222 | 211 |
| S21 | 402 | 20959 | 313 | 19 | 3 | 580008 | 0 | 0.0 | 204235 | 2267 | 2131 |

Table C.6: Results of $\text{CTSPAV}_{\text{SEC}}$ and $\text{CTSPAV}_{\text{Base}}$ for the Tight Problem Instances

| Instance name | CTSPAV variant | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SEC | | | | | | | Base | | | | | | |
| | Vehicle count | Total distance (m) | Vehicle count gap | Optimality gap (%) | Nodes explored | Wall time (s) | | Vehicle count | Total distance (m) | Vehicle count gap | Optimality gap (%) | Nodes explored | Wall time (s) | |
| | | | | | | MIP | Optimal count | | | | | | MIP | Optimal count |
| S0 | 5 | 961566 | 0 | 0.0 | 95266 | 388 | 82 | 5 | 961566 | 0 | 0.0 | 151291 | 533 | 90 |
| S1 | 3 | 619257 | 0 | 0.0 | 9643 | 97 | 89 | 3 | 619257 | 0 | 0.0 | 24230 | 326 | 323 |
| S2 | 5 | 1246019 | 0 | 0.0 | 13952 | 277 | 203 | 5 | 1246019 | 0 | 0.0 | 21299 | 917 | 902 |
| S3 | 5 | 1192722 | 1 | 20.0 | 178946 | 7201 | - | 5 | 1192722 | 1 | 19.9 | 241540 | 7201 | - |
| S4 | 5 | 1187914 | 0 | 0.0 | 400941 | 2668 | 187 | 5 | 1187914 | 0 | 0.0 | 17315 | 406 | 225 |
| S5 | 3 | 676142 | 0 | 0.0 | 3023 | 13 | 5 | 3 | 676142 | 0 | 0.0 | 4393 | 14 | 6 |
| S6 | 6 | 1503404 | 0 | 0.0 | 14190 | 284 | 284 | 6 | 1503404 | 0 | 0.0 | 73967 | 1653 | 1567 |
| S7 | 5 | 1345009 | 0 | 0.0 | 216353 | 3000 | 2352 | 5 | 1345009 | 0 | 0.0 | 243780 | 2824 | 1953 |
| S8 | 5 | 1310231 | 0 | 0.0 | 1459 | 22 | 21 | 5 | 1310231 | 0 | 0.0 | 3948 | 46 | 44 |
| S9 | 6 | 1094536 | 1 | 16.6 | 152214 | 7202 | - | 6 | 1094536 | 1 | 16.6 | 193079 | 7201 | - |
| S10 | 4 | 805606 | 0 | 0.0 | 16966 | 236 | 226 | 4 | 805606 | 0 | 0.0 | 9222 | 84 | 80 |
| S11 | 4 | 819652 | 0 | 0.0 | 9997 | 168 | 150 | 4 | 819652 | 0 | 0.0 | 12619 | 210 | 197 |
| S12 | 4 | 837723 | 0 | 0.0 | 161991 | 665 | 99 | 4 | 837723 | 0 | 0.0 | 155274 | 554 | 157 |
| S13 | 4 | 914708 | 0 | 0.0 | 94311 | 1553 | 1301 | 4 | 914708 | 0 | 0.0 | 304917 | 5579 | 5231 |
| S14 | 5 | 1450697 | 0 | 0.0 | 1250 | 44 | 31 | 5 | 1450697 | 0 | 0.0 | 14449 | 39 | 7 |
| S15 | 5 | 1613836 | 0 | 0.0 | 3338 | 24 | 12 | 5 | 1613836 | 0 | 0.0 | 1600 | 19 | 11 |
| S16 | 5 | 1220586 | 0 | 0.0 | 3471 | 78 | 72 | 5 | 1220586 | 0 | 0.0 | 1348 | 54 | 52 |
| S17 | 5 | 1252397 | 0 | 0.0 | 7338 | 48 | 32 | 5 | 1252397 | 0 | 0.0 | 10428 | 76 | 55 |
| S18 | 6 | 1452716 | 0 | 0.0 | 26594 | 278 | 268 | 6 | 1452716 | 0 | 0.0 | 22340 | 375 | 374 |
| S19 | 4 | 1030225 | 0 | 0.0 | 553629 | 4371 | 1324 | 4 | 1030225 | 0 | 0.0 | 173744 | 1736 | 326 |
| S20 | 4 | 1144849 | 0 | 0.0 | 9894 | 199 | 188 | 4 | 1144849 | 0 | 0.0 | 22515 | 504 | 500 |
| S21 | 3 | 580008 | 0 | 0.0 | 138098 | 2078 | 2042 | 3 | 580008 | 1 | 33.3 | 886186 | 7201 | - |

Tables C.7, C.8, and C.9 list results of the CTSPAV column-generation heuristic from Chapter 5 for every large, medium, and tight instance respectively. Their first columns show the instance names, followed by three columns that show the number of columns (mini routes) generated, the final vehicle count, and the total travel distance results for every instance. The following two columns display the absolute gap of its vehicle-count results and the optimality gap of its best incumbent solution. Since the heuristic does not utilize all the feasible mini routes, it has to use the optimal LP solutions of $\text{RMP}_{\text{CTSPAV}}$ to derive the primal lower bounds for these gap calculations. The final three columns show the percentage difference between the column count, the vehicle count, and the total distance of the heuristic relative to those of $\text{CTSPAV}_{\text{Hybrid}}$.

Table C.7: Results of CTSPAV Column-Generation Heuristic for Large Problem Instances

| Instance name | Column count | Vehicle count | Total distance (m) | Vehicle count gap | Optimality gap (%) | Percentage difference | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | Column count | Vehicle count | Total distance |
| L0 | 2231 | 3 | 647661 | 2 | 66.5 | -40% | 0% | -0.75% |
| L1 | 901 | 3 | 463065 | 2 | 66.6 | -18% | 0% | 0.00% |
| L2 | 8713 | 4 | 817348 | 3 | 74.9 | -83% | 0% | 0.05% |
| L3 | 9347 | 4 | 841180 | 3 | 74.8 | -85% | 0% | -0.30% |
| L4 | 7253 | 4 | 813018 | 3 | 74.8 | -85% | 0% | -0.14% |
| L5 | 960 | 3 | 512675 | 2 | 66.6 | -40% | 0% | 0.00% |
| L6 | 6330 | 4 | 955285 | 3 | 74.8 | -68% | 0% | 1.19% |
| L7 | 5087 | 4 | 888490 | 3 | 74.8 | -76% | 0% | 0.35% |
| L8 | 4902 | 4 | 844674 | 3 | 74.8 | -67% | 0% | 0.00% |
| L9 | 13892 | 5 | 737361 | 4 | 79.9 | -93% | 0% | -0.22% |
| L10 | 2884 | 3 | 555102 | 2 | 66.5 | -50% | 0% | 0.05% |
| L11 | 5659 | 3 | 570036 | 2 | 66.5 | -69% | 0% | 0.88% |
| L12 | 2116 | 3 | 581863 | 2 | 66.5 | -42% | 0% | 0.00% |
| L13 | 3106 | 3 | 624843 | 2 | 66.5 | -55% | 0% | 0.01% |
| L14 | 9539 | 4 | 949361 | 3 | 74.8 | -87% | 0% | -0.66% |
| L15 | 8161 | 4 | 1108007 | 3 | 74.8 | -93% | 0% | 0.80% |
| L16 | 3513 | 4 | 847394 | 3 | 74.8 | -62% | 0% | 0.37% |
| L17 | 2886 | 4 | 862155 | 3 | 74.8 | -55% | 0% | 0.11% |
| L18 | 2912 | 4 | 914762 | 3 | 74.8 | -34% | 0% | 0.49% |
| L19 | 6278 | 3 | 698599 | 2 | 74.9 | -82% | 33% | 0.27% |
| L20 | 8291 | 3 | 779684 | 2 | 66.5 | -86% | 0% | -1.40% |
| L21 | 1397 | 2 | 457911 | 1 | 49.9 | -71% | 0% | -0.01% |

Table C.8: Results of CTSPAV Column-Generation Heuristic for Medium Problem Instances

| Instance name | Column count | Vehicle count | Total distance (m) | Vehicle count gap | Optimality gap (%) | Percentage difference | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | Column count | Vehicle count | Total distance |
| M0 | 1664 | 2 | 481141 | 1 | 49.9 | -49% | 0% | -0.19% |
| M1 | 2643 | 3 | 605515 | 2 | 66.5 | -71% | 0% | -0.17% |
| M2 | 4349 | 3 | 846579 | 2 | 66.5 | -86% | 0% | 0.75% |
| M3 | 5461 | 3 | 668490 | 2 | 66.5 | -82% | 0% | 1.07% |
| M4 | 3556 | 3 | 535195 | 2 | 66.6 | -87% | 0% | 0.04% |
| M5 | 464 | 2 | 333048 | 1 | 49.9 | -8% | 0% | 0.00% |
| M6 | 6217 | 3 | 657988 | 2 | 66.5 | -87% | 0% | 0.36% |
| M7 | 2081 | 3 | 595519 | 2 | 66.5 | -40% | 0% | 0.00% |
| M8 | 3728 | 3 | 689147 | 2 | 66.5 | -66% | 0% | -0.21% |
| M9 | 6545 | 3 | 489997 | 2 | 66.6 | -88% | 0% | 0.01% |
| M10 | 6938 | 3 | 719639 | 2 | 66.5 | -94% | 0% | 0.03% |
| M11 | 2142 | 2 | 602968 | 1 | 49.9 | -63% | 0% | -0.40% |
| M12 | 1198 | 2 | 417175 | 1 | 49.9 | -36% | 0% | 0.00% |
| M13 | 4821 | 3 | 652724 | 2 | 66.5 | -83% | 0% | 0.17% |
| M14 | 1712 | 2 | 401064 | 1 | 49.9 | -47% | 0% | 0.08% |
| M15 | 4122 | 3 | 627967 | 2 | 74.9 | -72% | 33% | -1.07% |
| M16 | 1849 | 3 | 599126 | 2 | 66.5 | -53% | 0% | 0.07% |
| M17 | 964 | 2 | 490178 | 1 | 49.9 | -30% | 0% | 0.00% |
| M18 | 528 | 2 | 347259 | 1 | 49.9 | -30% | 0% | 0.00% |
| M19 | 914 | 2 | 339073 | 1 | 49.9 | -71% | 0% | 0.00% |
| M20 | 2153 | 3 | 551547 | 2 | 66.5 | -51% | 0% | 0.00% |
| M21 | 1172 | 3 | 620764 | 2 | 66.5 | -31% | 0% | 0.00% |
| M22 | 4527 | 3 | 683612 | 2 | 66.5 | -77% | 0% | 0.16% |
| M23 | 3416 | 3 | 556522 | 2 | 66.5 | -72% | 0% | -0.09% |
| M24 | 4949 | 3 | 588191 | 2 | 66.5 | -79% | 0% | 0.04% |
| M25 | 3969 | 3 | 596412 | 2 | 66.5 | -79% | 0% | 0.16% |
| M26 | 1043 | 3 | 445952 | 2 | 66.6 | -33% | 0% | 0.09% |
| M27 | 2336 | 3 | 712881 | 2 | 66.5 | -53% | 0% | 0.02% |
| M28 | 1810 | 2 | 394323 | 1 | 49.9 | -49% | 0% | 0.00% |
| M29 | 6028 | 3 | 729149 | 2 | 66.5 | -84% | 0% | -0.38% |

Table C.9: Results of CTSPAV Column-Generation Heuristic for Tight Problem Instances

| Instance name | Column count | Vehicle count | Total distance (m) | Vehicle count gap | Optimality gap (%) | Percentage difference | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | Column count | Vehicle count | Total distance |
| S0 | 359 | 5 | 961566 | 2 | 39.9 | -4% | 0% | 0.00% |
| S1 | 267 | 3 | 619257 | 1.97 | 65.4 | 0% | 0% | 0.00% |
| S2 | 813 | 5 | 1246019 | 2 | 39.9 | -16% | 0% | 0.00% |
| S3 | 969 | 5 | 1192722 | 2.92 | 58.3 | -24% | 0% | 0.00% |
| S4 | 840 | 5 | 1187914 | 1 | 20.0 | -30% | 0% | 0.05% |
| S5 | 291 | 3 | 676142 | 1 | 33.3 | -4% | 0% | 0.00% |
| S6 | 633 | 6 | 1503404 | 2 | 33.3 | -10% | 0% | 0.00% |
| S7 | 575 | 5 | 1345009 | 2 | 49.9 | -16% | 20% | -1.19% |
| S8 | 502 | 5 | 1310231 | 1 | 19.9 | -14% | 0% | 0.00% |
| S9 | 1273 | 6 | 1094536 | 3 | 49.9 | -24% | 0% | 0.13% |
| S10 | 421 | 4 | 805606 | 2 | 49.9 | -2% | 0% | 0.00% |
| S11 | 744 | 4 | 819652 | 1 | 25.0 | -11% | 0% | 0.01% |
| S12 | 377 | 4 | 837723 | 2 | 49.9 | -4% | 0% | 0.00% |
| S13 | 461 | 4 | 914708 | 2 | 49.9 | -9% | 0% | 0.12% |
| S14 | 876 | 5 | 1450697 | 1.62 | 32.2 | -17% | 0% | 0.01% |
| S15 | 1072 | 5 | 1613836 | 1 | 19.9 | -62% | 0% | 0.01% |
| S16 | 502 | 5 | 1220586 | 2 | 39.9 | -5% | 0% | 0.00% |
| S17 | 453 | 5 | 1252397 | 2 | 39.9 | -9% | 0% | 0.00% |
| S18 | 383 | 6 | 1452716 | 2 | 33.3 | -2% | 0% | 0.00% |
| S19 | 762 | 4 | 1030225 | 1.50 | 37.4 | -40% | 0% | 0.15% |
| S20 | 934 | 4 | 1144849 | 1 | 24.9 | -39% | 0% | 0.00% |
| S21 | 305 | 3 | 580008 | 2 | 66.5 | -3% | 0% | 0.00% |

# BIBLIOGRAPHY


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, ISSN 0377-2217, URL http://dx.doi.org/https://doi.org/10.1016/j.ejor.2012.05.028.

Agatz NA, Erera AL, Savelsbergh MW, Wang X (2011) Dynamic ride-sharing: A simulation study in metro Atlanta. *Transportation Research Part B: Methodological* 45(9):1450 – 1464, ISSN 0191-2615, URL http://dx.doi.org/https://doi.org/10.1016/j.trb.2011.05.017, select Papers from the 19th ISTTT.

Alazzawi S, Hummel M, Kordt P, Sickenberger T, Wieseotte C, Wohak O (2018) Simulating the impact of shared, autonomous vehicles on urban mobility – A case study of Milan. Wießner E, Lücken L, Hilbrich R, Flötteröd YP, Erdmann J, Bieker-Walz L, Behrisch M, eds., *SUMO 2018- Simulating Autonomous and Intermodal Transport Systems*, volume 2 of *EPiC Series in Engineering*, 94–110 (EasyChair), ISSN 2516-2330, URL http://dx.doi.org/10.29007/2n4h.

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, ISSN 0027-8424, URL http://dx.doi.org/10.1073/pnas.1611675114.

Arning K, Ziefle M, Muehlhans H (2013) Join the ride! User requirements and interface design guidelines for a commuter carpooling platform. Marcus A, ed., *Design, User Experience, and Usability. User Experience in Novel Technological Environments*, 10–19 (Berlin, Heidelberg: Springer Berlin Heidelberg), ISBN 978-3-642-39238-2.

Arthur D, Vassilvitskii S (2007) $k$-means++: The advantages of careful seeding. *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1027–1035, SODA '07 (USA: Society for Industrial and Applied Mathematics), ISBN 9780898716245.

Ascheuer N, Fischetti M, Grötschel M (2000) A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks* 36(2):69–79, ISSN 0028-3045, URL http://dx.doi.org/10.1002/1097-0037(200009)36:2<69::AID-NET1>3.0.CO;2-Q.

Ascheuer N, Fischetti M, Grötschel M (2001) Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming* 90(3):475–506, ISSN 1436-4646, URL http://dx.doi.org/10.1007/PL00011432.

Balas E, Fischetti M, Pulleyblank WR (1995) The precedence-constrained asymmetric traveling salesman polytope. *Mathematical Programming* 68(1):241–265, ISSN 1436-4646, URL http://dx.doi.org/10.1007/BF01585767.

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, URL http://dx.doi.org/10.1287/opre.1030.0106.

Bard JF, Kontoravdis G, Yu G (2002) A branch-and-cut procedure for the vehicle routing problem with time windows. *Transportation Science* 36(2):250–269, URL http://dx.doi.org/10.1287/trsc.36.2.250.565.

Basciftci B, Van Hentenryck P (2020) Bilevel optimization for on-demand multimodal transit systems. Hebrard E, Musliu N, eds., *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, 52–68 (Cham: Springer International Publishing), ISBN 978-3-030-58942-4.

Bastert O, Hummel B, de Vries S (2010) A generalized Wedelin heuristic for integer programming. *INFORMS Journal on Computing* 22(1):93–107, URL http://dx.doi.org/10.1287/ijoc.1090.0328.

Beasley JE, Christofides N (1989) An algorithm for the resource constrained shortest path problem. *Networks* 19(4):379–394, URL http://dx.doi.org/10.1002/net.3230190402.

Bent RW, Van Hentenryck P (2004) Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research* 52(6):977–987, URL http://dx.doi.org/10.1287/opre.1040.0124.

Bodin L, Sexton T (1986) The multi-vehicle subscriber dial-a-ride problem. *TIMS Studies in Management Science* 22:73–86.

Boland N, Dethridge J, Dumitrescu I (2006) Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters* 34(1):58 – 68, ISSN 0167-6377, URL http://dx.doi.org/https://doi.org/10.1016/j.orl.2004.11.011.

Bongiovanni C, Kaspi M, Geroliminis N (2019) The electric autonomous dial-a-ride problem. *Transportation Research Part B: Methodological* 122:436 – 456, ISSN 0191-2615, URL http://dx.doi.org/https://doi.org/10.1016/j.trb.2019.03.004.

Borndörfer R, Grötschel M, Löbel A (2001) Scheduling duties by adaptive column generation. ZIB-Report 01-02. Konrad-Zuse-Zentrum für Informationstechnik Berlin.

Bräysy O, Gendreau M (2005) Vehicle routing problem with time windows, Part II: Metaheuristics. *Transportation Science* 39(1):119–139, URL http://dx.doi.org/10.1287/trsc.1030.0057.

Buliung RN, Soltys K, Bui R, Habel C, Lanyon R (2010) Catching a ride on the information superhighway: Toward an understanding of internet-based carpool formation and use. *Transportation* 37(6):849–873, ISSN 1572-9435, URL http://dx.doi.org/10.1007/s11116-010-9266-0.

Buliung RN, Soltys K, Habel C, Lanyon R (2009) Driving factors behind successful carpool formation and use. *Transportation Research Record* 2118(1):31–38, URL http://dx.doi.org/10.3141/2118-05.

Chabrier A (2006) Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research* 33(10):2972 – 2990, ISSN 0305-0548, URL http://dx.doi.org/https://doi.org/10.1016/j.cor.2005.02.029, part Special Issue: Constraint Programming.

Chesley K (2017) Good news, bad news about parking on campus. Stanford News, URL https://news.stanford.edu/2017/01/06/good-news-bad-news-parking-campus/.

Cookson G (2018) INRIX global traffic scorecard. INRIX Research.

Cordeau JF (2006) A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research* 54(3):573–586, URL http://dx.doi.org/10.1287/opre.1060.0283.

Cordeau JF, Desaulniers G, Desrosiers J, Solomon MM, Soumis F (2002) VRP with time windows. Toth P, Vigo D, eds., *The Vehicle Routing Problem*, chapter 7, 157–193 (Philadelphia, PA, USA: SIAM monographs on discrete mathematics and applications), URL http://dx.doi.org/10.1137/1.9780898718515.ch7.

Cordeau JF, Laporte G (2003a) The Dial-a-Ride Problem (DARP): Variants, modeling issues and algorithms. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies* 1(2):89–101, ISSN 1619-4500, URL http://dx.doi.org/10.1007/s10288-002-0009-8.

Cordeau JF, Laporte G (2003b) A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological* 37(6):579 – 594, ISSN 0191-2615, URL http://dx.doi.org/https://doi.org/10.1016/S0191-2615(02)00045-0.

Cordeau JF, Laporte G (2007) The dial-a-ride problem: Models and algorithms. *Annals of Operations Research* 153(1):29–46, ISSN 1572-9338, URL http://dx.doi.org/10.1007/s10479-007-0170-8.

Correia G, Viegas JM (2011) Carpooling and carpool clubs: Clarifying concepts and assessing value enhancement possibilities through a stated preference web survey in Lisbon, Portugal. *Transportation Research Part A: Policy and Practice* 45(2):81 – 90, ISSN 0965-8564, URL http://dx.doi.org/https://doi.org/10.1016/j.tra.2010.11.001.

Dantzig G, Fulkerson R, Johnson S (1954) Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America* 2(4):393–410, URL http://dx.doi.org/10.1287/opre.2.4.393.

Dantzig GB, Wolfe P (1960) Decomposition principle for linear programs. *Operations Research* 8(1):101–111, URL http://dx.doi.org/10.1287/opre.8.1.101.

Desaulniers G, Lessard F, Hadjar A (2008) Tabu search, partial elementarity, and generalized $k$-path inequalities for the vehicle routing problem with time windows. *Transportation Science* 42(3):387–404, URL http://dx.doi.org/10.1287/trsc.1070.0223.

Desrochers M (1988) An algorithm for the shortest path problem with resource constraints. Technical Report G-88-27, Les Cahiers du GERAD, Montreal (Quebec), Canada.

Desrochers M, Desrosiers J, Solomon MM (1992) A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research* 40(2):342–354, URL http://dx.doi.org/10.1287/opre.40.2.342.

Desrochers M, Laporte G (1991) Improvements and extensions to the Miller–Tucker–Zemlin subtour elimination constraints. *Operations Research Letters* 10(1):27 – 36, ISSN 0167-6377, URL http://dx.doi.org/https://doi.org/10.1016/0167-6377(91)90083-2.

Desrochers M, Soumis F (1988) A generalized permanent labelling algorithm for the shortest path problem with time windows. *INFOR: Information Systems and Operational Research* 26(3):191–212, URL http://dx.doi.org/10.1080/03155986.1988.11732063.

Desrosiers J, Pelletier P, Soumis F (1983) Plus court chemin avec contraintes d'horaires. *RAIRO* -

*Operations Research - Recherche Opérationnelle* 17(4):357–377, URL http://eudml.org/doc/104840.

Desrosiers J, Soumis F, Desrochers M (1984) Routing with time windows by column generation. *Networks* 14(4):545–565, URL http://dx.doi.org/10.1002/net.3230140406.

Di Puglia Pugliese L, Guerriero F (2013a) Dynamic programming approaches to solve the shortest path problem with forbidden paths. *Optimization Methods and Software* 28(2):221–255, URL http://dx.doi.org/10.1080/10556788.2011.630077.

Di Puglia Pugliese L, Guerriero F (2013b) Shortest path problem with forbidden paths: The elementary version. *European Journal of Operational Research* 227(2):254 – 267, ISSN 0377-2217, URL http://dx.doi.org/https://doi.org/10.1016/j.ejor.2012.11.010.

Dia H, Javanshour F (2017) Autonomous shared mobility-on-demand: Melbourne pilot simulation study. *Transportation Research Procedia* 22:285 – 296, ISSN 2352-1465, URL http://dx.doi.org/https://doi.org/10.1016/j.trpro.2017.03.035, 19th EURO Working Group on Transportation Meeting, EWGT2016, 5-7 September 2016, Istanbul, Turkey.

Drexl M (2013) A note on the separation of subtour elimination constraints in elementary shortest path problems. *European Journal of Operational Research* 229(3):595 – 598, ISSN 0377-2217, URL http://dx.doi.org/https://doi.org/10.1016/j.ejor.2013.03.009.

Dror M (1994) Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research* 42(5):977–978, URL http://dx.doi.org/10.1287/opre.42.5.977.

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

Epstein JD (2018) Parking spaces become more elusive as downtown Buffalo booms. The Buffalo News, URL https://buffalonews.com/2018/11/23/brother-can-you-spare-a-parking-space/.

Farhan J, Chen TD (2018) Impact of ridesharing on operational efficiency of shared autonomous electric vehicle fleet. *Transportation Research Part C: Emerging Technologies* 93:310 – 321, ISSN 0968-090X, URL http://dx.doi.org/https://doi.org/10.1016/j.trc.2018.04.022.

Farley AA (1990) A note on bounding a class of linear programming problems, including cutting stock problems. *Operations Research* 38(5):922–923, URL http://dx.doi.org/10.1287/opre.38.5.922.

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, URL http://dx.doi.org/10.1002/net.20033.

Firat M, Woeginger GJ (2011) Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh. *Operations Research Letters* 39(1):32 – 35, ISSN 0167-6377, URL http://dx.doi.org/https://doi.org/10.1016/j.orl.2010.11.004.

Fischetti M, Toth P (1997) A polyhedral approach to the asymmetric traveling salesman problem. *Management Science* 43(11):1520–1536, URL http://dx.doi.org/10.1287/mnsc.43.11.1520.

Friedrich B (2015) Verkehrliche wirkung autonomer fahrzeuge. Maurer M, Gerdes JC, Lenz B, Winner H, eds., *Autonomes Fahren: Technische, rechtliche und gesellschaftliche Aspekte*, 331–350 (Berlin, Heidelberg: Springer Berlin Heidelberg), ISBN 978-3-662-45854-9, URL http://dx.doi.org/10.1007/978-3-662-45854-9_16.

Gomory RE (1960) An algorithm for the mixed integer problem. Technical Report RM-2597, The RAND Corporation, Santa Monica, California, USA.

Gomory RE, Hu TC (1961) Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics* 9(4):551–570, URL http://dx.doi.org/10.1137/0109047.

Gouveia L, Pires JM (1999) The asymmetric travelling salesman problem and a reformulation of the Miller–Tucker–Zemlin constraints. *European Journal of Operational Research* 112(1):134 – 146, ISSN 0377-2217, URL http://dx.doi.org/https://doi.org/10.1016/S0377-2217(97)00358-5.

Grötschel M, Padberg M (1985) Polyhedral theory. Lawler E, Lenstra J, Rinnooy Kan A, Shmoys D, eds., *The Traveling Salesman Problem*, chapter 8, 251–305, A Wiley-Interscience publication (John Wiley & Sons, Incorporated), ISBN 9780471904137, URL https://books.google.com/books?id=EPFQAAAAMAAJ.

Grötschel M, Padberg MW (1975) Partial linear characterizations of the asymmetric travelling salesman polytope. *Mathematical Programming* 8(1):378–381, ISSN 1436-4646, URL http://dx.doi.org/10.1007/BF01580454.

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, URL http://dx.doi.org/10.1287/trsc.2014.0531.

Gurumurthy KM, Kockelman KM (2018) Analyzing the dynamic ride-sharing potential for shared autonomous vehicle fleets using cellphone data from Orlando, Florida. *Computers, Environment and Urban Systems* 71:177 – 185, ISSN 0198-9715, URL http://dx.doi.org/https://doi.org/10.1016/j.compenvurbsys.2018.05.008.

Hasan MH, Van Hentenryck P (2020) The flexible and real-time commute trip sharing problems. *Constraints* 25(3):160–179, ISSN 1572-9354, URL http://dx.doi.org/10.1007/s10601-020-09310-5.

Hasan MH, Van Hentenryck P (2021a) The benefits of autonomous vehicles for community-based trip sharing. *Transportation Research Part C: Emerging Technologies* 124:102929, ISSN 0968-090X, URL http://dx.doi.org/https://doi.org/10.1016/j.trc.2020.102929.

Hasan MH, Van Hentenryck P (2021b) Commuting with autonomous vehicles: A branch-and-cut algorithm with redundant modeling. *arXiv e-prints* arXiv:2101.01072.

Hasan MH, Van Hentenryck P, Budak C, Chen J, Chaudhry C (2018) Community-based trip sharing for urban commuting. McIlraith S, Weinberger K, eds., *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 6589–6597, AAAI-18 (Palo Alto, California, USA: AAAI Press).

Hasan MH, Van Hentenryck P, Legrain A (2020) The commute trip-sharing problem. *Transportation Science* 54(6):1640–1675, URL http://dx.doi.org/10.1287/trsc.2019.0969.

Haugland D, Ho SC (2010) Feasibility testing for dial-a-ride problems. Chen B, ed., *Algorithmic Aspects in Information and Management*, 170–179 (Berlin, Heidelberg: Springer Berlin Heidelberg), ISBN 978-3-642-14355-7.

Helsgaun K (2000) An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research* 126(1):106–130, ISSN 0377-2217, URL http://dx.doi.org/https://doi.org/10.1016/S0377-2217(99)00284-2.

Hunsaker B, Savelsbergh M (2002) Efficient feasibility testing for dial-a-ride problems. *Operations Research Letters* 30(3):169 – 173, ISSN 0167-6377, URL http://dx.doi.org/https://doi.org/10.1016/S0167-6377(02)00120-7.

Irnich S, Desaulniers G (2005) Shortest path problems with resource constraints. Desaulniers G, Desrosiers J, Solomon MM, eds., *Column Generation*, 33–65 (Boston, MA: Springer US), ISBN 978-0-387-25486-9, URL http://dx.doi.org/10.1007/0-387-25486-2_2.

Irnich S, Villeneuve D (2006) The shortest-path problem with resource constraints and $k$-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing* 18(3):391–406, URL http://dx.doi.org/10.1287/ijoc.1040.0117.

Jaw JJ, Odoni AR, Psaraftis HN, Wilson NH (1986) A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B: Methodological* 20(3):243 – 257, ISSN 0191-2615, URL http://dx.doi.org/https://doi.org/10.1016/0191-2615(86)90020-2.

Kallehauge B, Boland N, Madsen OB (2007) Path inequalities for the vehicle routing problem with time windows. *Networks* 49(4):273–293, URL http://dx.doi.org/10.1002/net.20178.

Kallehauge B, Larsen J, Madsen OB (2006) Lagrangian duality applied to the vehicle routing problem with time windows. *Computers & Operations Research* 33(5):1464 – 1487, ISSN 0305-0548, URL http://dx.doi.org/https://doi.org/10.1016/j.cor.2004.11.002.

Kohl N, Desrosiers J, Madsen OBG, Solomon MM, Soumis F (1999) 2-path cuts for the vehicle routing problem with time windows. *Transportation Science* 33(1):101–116, URL http://dx.doi.org/10.1287/trsc.33.1.101.

Kohl N, Madsen OBG (1997) An optimization algorithm for the vehicle routing problem with time windows based on Lagrangian relaxation. *Operations Research* 45(3):395–406, URL http://dx.doi.org/10.1287/opre.45.3.395.

Langevin A, Soumis F, Desrosiers J (1990) Classification of travelling salesman problem formulations. *Operations Research Letters* 9(2):127 – 132, ISSN 0167-6377, URL http://dx.doi.org/https://doi.org/10.1016/0167-6377(90)90052-7.

Lasdon LS (1970) *Optimization theory for large systems* (MacMillan).

Li J, Embry PM, Mattingly SP, Sadabadi KF, Rasmidatta I, Burris MW (2007) Who chooses to carpool and why?: Examination of Texas carpoolers. *Transportation Research Record: Journal of the Transportation Research Board* (2021):110–117.

Liberti L (2004) Reduction constraints for the global optimization of NLPs. *International Transactions in Operational Research* 11(1):33–41, URL http://dx.doi.org/https://doi.org/10.1111/j.1475-3995.2004.00438.x.

Lin S, Kernighan BW (1973) An effective heuristic algorithm for the traveling-salesman problem. *Operations Research* 21(2):498–516, URL http://dx.doi.org/10.1287/opre.21.2.498.

Lloyd S (1982) Least squares quantization in PCM. *IEEE Transactions on Information Theory*

28(2):129–137, ISSN 1557-9654, URL http://dx.doi.org/10.1109/TIT.1982.1056489.

Lübbecke ME, Desrosiers J (2005) Selected topics in column generation. *Operations Research* 53(6):1007–1023, URL http://dx.doi.org/10.1287/opre.1050.0234.

Ma J, Li X, Zhou F, Hao W (2017) Designing optimal autonomous vehicle sharing and reservation systems: A linear programming approach. *Transportation Research Part C: Emerging Technologies* 84:124 – 141, ISSN 0968-090X, URL http://dx.doi.org/https://doi.org/10.1016/j.trc.2017.08.022.

Mahéo A, Kilby P, Van Hentenryck P (2019) Benders decomposition for the design of a hub and shuttle public transit system. *Transportation Science* 53(1):77–88, URL http://dx.doi.org/10.1287/trsc.2017.0756.

Martinez LM, Viegas JM (2017) Assessing the impacts of deploying a shared self-driving urban mobility system: An agent-based model applied to the city of Lisbon, Portugal. *International Journal of Transportation Science and Technology* 6(1):13 – 27, ISSN 2046-0430, URL http://dx.doi.org/https://doi.org/10.1016/j.ijtst.2017.05.005, Connected and Automated Vehicles: Effects on Traffic, Mobility and Urban Design.

McKenzie B (2015) Who drives to work? Commuting by automobile in the United States: 2013. American Community Survey Reports. ACS-32. U.S. Census Bureau, Washington, DC.

Mena-Oreja J, Gozalvez J, Sepulcre M (2018) Effect of the configuration of platooning maneuvers on the traffic flow under mixed traffic scenarios. *2018 IEEE Vehicular Networking Conference (VNC)*, 1–4, ISSN 2157-9865, URL http://dx.doi.org/10.1109/VNC.2018.8628381.

Milakis D, van Arem B, van Wee B (2017) Policy and society related implications of automated driving: A review of literature and directions for future research. *Journal of Intelligent Transportation Systems* 21(4):324–348, URL http://dx.doi.org/10.1080/15472450.2017.1291351.

Miller CE, Tucker AW, Zemlin RA (1960) Integer programming formulation of traveling salesman problems. *J. ACM* 7(4):326–329, ISSN 0004-5411, URL http://dx.doi.org/10.1145/321043.321046.

Mourad A, Puchinger J, Chu C (2019) A survey of models and algorithms for optimizing shared mobility. *Transportation Research Part B: Methodological* 123:323 – 346, ISSN 0191-2615, URL http://dx.doi.org/https://doi.org/10.1016/j.trb.2019.02.003.

Naddef D, Rinaldi G (2001) Branch-and-cut algorithms for the capacitated VRP. *The Vehicle Routing Problem*, 53–84 (USA: Society for Industrial and Applied Mathematics), ISBN 0898714982.

Narayanan S, Chaniotakis E, Antoniou C (2020) Shared autonomous vehicle services: A comprehensive review. *Transportation Research Part C: Emerging Technologies* 111:255 – 293, ISSN 0968-090X, URL http://dx.doi.org/https://doi.org/10.1016/j.trc.2019.12.008.

NYC Taxi & Limousine Commission (2020) TLC trip record data. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page, accessed: 2020-11-20.

Olia A, Razavi S, Abdulhai B, Abdelgawad H (2018) Traffic capacity implications of automated vehicles mixed with regular vehicles. *Journal of Intelligent Transportation Systems* 22(3):244–262, URL http://dx.doi.org/10.1080/15472450.2017.1404680.

Padberg M, Rinaldi G (1990) An efficient algorithm for the minimum capacity cut problem. *Mathematical Programming* 47(1):19–36, ISSN 1436-4646, URL http://dx.doi.org/10.1007/BF01580850.

Padberg M, Rinaldi G (1991) A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review* 33(1):60–100, URL http://dx.doi.org/10.1137/1033004.

Pinto HK, Hyland MF, Mahmassani HS, Verbas IÖ (2020) Joint design of multimodal transit networks and shared autonomous mobility fleets. *Transportation Research Part C: Emerging Technologies* 113:2 – 20, ISSN 0968-090X, URL http://dx.doi.org/https://doi.org/10.1016/j.trc.2019.06.010, 23rd International Symposium on Transportation and Traffic Theory (ISTTT 23).

Poulenez-Donovan CJ, Ulberg C (1994) Seeing the trees and missing the forest: Qualitative versus quantitative research findings in a model transportation demand management program evaluation. *Transportation Research Record* (1459):1–6.

Raghunathan AU, Bergman D, Hooker JN, Serra T, Kobori S (2018a) The integrated last-mile transportation problem (ILMTP). de Weerdt M, Koenig S, Röger G, Spaan MTJ, eds., *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling, ICAPS 2018, Delft, The Netherlands, June 24-29, 2018*, 388–398 (AAAI Press), URL https://aaai.org/ocs/index.php/ICAPS/ICAPS18/paper/view/17720.

Raghunathan AU, Bergman D, Hooker JN, Serra T, Kobori S (2018b) Seamless multimodal transportation scheduling. *arXiv e-prints* arXiv:1807.09676.

Richardson A, Young W (1981) Spatial relationships between carpool members' trip ends. *Transportation Research Record* (823):1–7.

Ritzinger U, Puchinger J, Hartl RF (2016) Dynamic programming based metaheuristics for the dial-a-ride problem. *Annals of Operations Research* 236(2):341–358, ISSN 1572-9338, URL http://dx.doi.org/10.1007/s10479-014-1605-7.

Ropke S, Cordeau JF (2006) *Heuristic and exact algorithms for vehicle routing problems*. Ph.D. thesis, University of Copenhagen, Branch-and-cut-and-price for the pickup and delivery problem with time windows.

Ropke S, Cordeau JF (2009) Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science* 43(3):267–286, URL http://dx.doi.org/10.1287/trsc.1090.0272.

Rousseau LM, Gendreau M, Feillet D (2007) Interior point stabilization for column generation. *Operations Research Letters* 35(5):660 – 668, ISSN 0167-6377, URL http://dx.doi.org/https://doi.org/10.1016/j.orl.2006.11.004.

Rousseau LM, Gendreau M, Pesant G, Focacci F (2004) Solving VRPTWs with constraint programming based column generation. *Annals of Operations Research* 130(1):199–216, ISSN 1572-9338, URL http://dx.doi.org/10.1023/B:ANOR.0000032576.73681.29.

Ruiz JP, Grossmann IE (2011) Using redundancy to strengthen the relaxation for the global optimization of MINLP problems. *Computers & Chemical Engineering* 35(12):2729 – 2740, ISSN 0098-1354, URL http://dx.doi.org/https://doi.org/10.1016/j.compchemeng.2011.01.035.

Ruland K, Rodin E (1997) The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computers & Mathematics with Applications* 33(12):1 – 13, ISSN 0898-1221, URL http://dx.doi.org/https://doi.org/10.1016/S0898-1221(97)00090-4.

Salazar M, Rossi F, Schiffer M, Onder CH, Pavone M (2018) On the interaction between autonomous mobility-on-demand and public transportation systems. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2262–2269, ISSN 2153-0017, URL http://dx.doi.org/10.1109/ITSC.2018.8569381.

Santi P, Resta G, Szell M, Sobolevsky S, Strogatz SH, Ratti C (2014) Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences* 111(37):13290–13294, ISSN 0027-8424, URL http://dx.doi.org/10.1073/pnas.1403657111.

Savelsbergh MWP (1985) Local search in routing problems with time windows. *Annals of Operations Research* 4(1):285–305, ISSN 1572-9338, URL http://dx.doi.org/10.1007/BF02022044.

Serra T, Raghunathan AU, Bergman D, Hooker J, Kobori S (2019) Last-mile scheduling under uncertainty. Rousseau LM, Stergiou K, eds., *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, 519–528 (Cham: Springer International Publishing), ISBN 978-3-030-19212-9.

Shaheen SA, Rodier CJ (2005) Travel effects of a suburban commuter carsharing service: Carlink case study. *Transportation Research Record* 1927(1):182–188, URL http://dx.doi.org/10.1177/0361198105192700121.

Shoup DC (1997) The high cost of free parking. *Journal of Planning Education and Research* 17(1):3–20, URL http://dx.doi.org/10.1177/0739456X9701700102.

Shoup DC (2006) Cruising for parking. *Transport Policy* 13(6):479 – 486, ISSN 0967-070X, URL http://dx.doi.org/https://doi.org/10.1016/j.tranpol.2006.05.005, parking.

Soteropoulos A, Berger M, Ciari F (2019) Impacts of automated vehicles on travel behaviour and land use: An international review of modelling studies. *Transport Reviews* 39(1):29–49, URL http://dx.doi.org/10.1080/01441647.2018.1523253.

Srour FJ, Agatz N, Oppen J (2018) Strategies for handling temporal uncertainty in pickup and delivery problems with time windows. *Transportation Science* 52(1):3–19, URL http://dx.doi.org/10.1287/trsc.2015.0658.

Stocker A, Shaheen S (2019) Shared automated vehicle (SAV) pilots and automated vehicle policy in the U.S.: Current and future developments. Meyer G, Beiker S, eds., *Road Vehicle Automation 5*, 131–147 (Cham: Springer International Publishing), ISBN 978-3-319-94896-6.

Taillard É, Badeau P, Gendreau M, Guertin F, Potvin JY (1997) A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science* 31(2):170–186, URL http://dx.doi.org/10.1287/trsc.31.2.170.

Talebpour A, Mahmassani HS (2016) Influence of connected and autonomous vehicles on traffic flow stability and throughput. *Transportation Research Part C: Emerging Technologies* 71:143 – 163, ISSN 0968-090X, URL http://dx.doi.org/https://doi.org/10.1016/j.trc.2016.07.007.

Tang J, Kong Y, Lau H, Ip AW (2010) A note on "Efficient feasibility testing for dial-a-ride problems". *Operations Research Letters* 38(5):405 – 407, ISSN 0167-6377, URL http://dx.doi.org/https://doi.org/10.1016/j.orl.2010.05.002.

Tarjan R (1972) Depth-first search and linear graph algorithms. *SIAM Journal on Computing* 1(2):146–160, URL http://dx.doi.org/10.1137/0201010.

Taylor E (2018) The elephant in the planning scheme: How cities still work around the dominance of parking space. The Conversation, URL http://theconversation.com/the-elephant-in-the-planning-scheme-how-cities-still-work-around-the-dominance-of-parking-space-87098.

Thomas M, Deepti T (2018) Reinventing carsharing as a modern and profitable service. The Intelligent Transportation Society of America 2018 Annual Meeting White Paper, URL https://ridecell.com/wp-content/uploads/White-Paper-Presentation_Reinventing-Carsharing-As-A-Modern-And-Profitable-Service.pdf.

Tientrakool P, Ho Y, Maxemchuk NF (2011) Highway capacity benefits from using vehicle-to-vehicle communication and sensors for collision avoidance. *2011 IEEE Vehicular Technology Conference (VTC Fall)*, 1–5, ISSN 1090-3038, URL http://dx.doi.org/10.1109/VETECF.2011.6093130.

Toth P, Vigo D (2002) *The Vehicle Routing Problem* (Society for Industrial and Applied Mathematics), URL http://dx.doi.org/10.1137/1.9780898718515.

Tsao HSJ, Lin DJ (1999) Spatial and temporal factors in estimating the potential of ride-sharing for demand reduction. Technical Report UCB-ITS-PRR-99-2, California Partners for Advanced Transportation Technology (PATH), Berkeley, California, USA.

Villeneuve D, Desaulniers G (2005) The shortest path problem with forbidden paths. *European Journal of Operational Research* 165(1):97 – 107, ISSN 0377-2217, URL http://dx.doi.org/https://doi.org/10.1016/j.ejor.2004.01.032.

Wang H, Cheu RL, Lee DH (2014) Intelligent taxi dispatch system for advance reservations. *Journal of Public Transportation* 17(3):115–128, URL http://dx.doi.org/http://doi.org/10.5038/2375-0901.17.3.8.

Wedelin D (1995) An algorithm for large scale 0–1 integer programming with application to airline crew scheduling. *Annals of Operations Research* 57(1):283–301, ISSN 1572-9338, URL http://dx.doi.org/10.1007/BF02099703.

Zhang W, Guhathakurta S (2017) Parking spaces in the age of shared autonomous vehicles: How much parking will we need and where? *Transportation Research Record* 2651(1):80–91, URL http://dx.doi.org/10.3141/2651-09.

Zhang W, Guhathakurta S, Fang J, Zhang G (2015) Exploring the impact of shared autonomous vehicles on urban parking demand: An agent-based simulation approach. *Sustainable Cities and Society* 19:34 – 45, ISSN 2210-6707, URL http://dx.doi.org/https://doi.org/10.1016/j.scs.2015.07.006.