ABSTRACT

Title of Dissertation: EXACT AND HEURISTIC METHODS FOR EMERGING VEHICLE ROUTING PROBLEMS

Eric Oden
Doctor of Philosophy, 2022

Dissertation Directed by: Professor Bruce Golden
Professor S. Raghavan
The Robert H. Smith School of Business

The rise of global supply chains and e-commerce in recent decades have intensified the relevance of the transportation industry to both the individual and the economy. With rising consumer expectations and slim profit margins, the various sectors within the transportation industry rely on the development of carefully designed routes to remain competitive. Despite the wealth of research on route design, and the responsiveness of the research community to practical considerations, there remain gaps between the work done in practice and that appearing in the literature. Correspondingly, the work in this dissertation is directly in response to conversations had with contacts from real-world companies within the transportation domain. We consider problems presented, verbally, by companies representing three distinct segments of the industry: freight routing, last-mile delivery, and on-demand passenger transport. Each problem is centered around an innovative strategy with the potential to dramatically disrupt its corresponding domain. First, we consider the Shared Truckload (STL) freight shipping model, an alternative to the

dominant Less-than-Truckload (LTL) model. Both models pool shipments from multiple customers into a single trailer, but, in the latter, consolidation is facilitated by a hub-and-spoke routing network, whereas, in the the former, freight moves directly from origin to destination. This strategy minimizes travel times and the risk of damage. We then investigate a novel strategy to facilitate last-mile, last-minute delivery, through coordinating a fleet of trucks and a fleet of smaller vehicles, referred to as shuttles. In order to accommodate requests which come in after trucks have been dispatched, shuttles are allowed to pick up packages from a depot and intercept trucks along their routes. This strategy can enable a shipper to make highly competitive service guarantees. Finally, we consider the emerging field of Urban Air Mobility (UAM), a vision of air taxis conveying passengers at lower altitudes throughout urban areas as an efficient alternative to gridlock traffic. In particular, we consider a UAM service company in the early stages of its development, where the chief goal is to maximize market share.

These innovations represent significant deviations from the status quo in their respective fields, and, thus, the existing research for each is slim, if existent. Therefore, we introduce precise mathematical formulations of each of the problems to the research community. We then develop both exact and heuristic approaches to solve the problems, and carry out extensive computational studies comparing the solution methodologies. Furthermore, for each of the problems, we offer a sensitivity analysis and managerial insights. Among our contributions are original algorithms based on solving a set-partitioning formulations via column generation, a highly successful paradigm for solving large linear programs. Among the advantages of this approach is the ability to include highly general route costs and constraints. We illustrate this expressiveness by demonstrating its application to each of the three highly distinct problems we consider.

EXACT AND HEURISTIC METHODS FOR NOVEL VEHICLE ROUTING PROBLEMS

by

Eric Oden

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2022

Acknowledgments

It is impossible to fully enumerate all of to whom I am indebted. However, I must thank my advisers, Dr. Bruce Golden and Dr. S. “Raghu” Raghavan for their expert guidance and support during the completion of this dissertation. Their insights and suggestions, drawn from deep experience with the world of transportation research, served as indispensable guides as I navigated the wealth of extant research and worked towards the results presented in this work. Throughout my studies, I was given the encouragement to explore and experiment with countless ideas, always able to rely on a prompt response after hitting any bump, fork, or brick wall. During our unfailingly collegial meetings, they helped disentangle my thoughts and ideas into clear, coherent research directions.

Furthermore, thanks to my advisers’ network of connections with industry practitioners, and the introductions they made, I was able work on cutting-edge problems with immediate relevance. Through Dr. Golden’s connection with RouteSmart Technologies, I was given an invaluable glimpse into how vehicle routing is done in practice. For that, I must also thank Dr. Damon Gulczynski and Dr. Larry Levy, whose welcome and guidance made my experience with RouteSmart not only edifying but deeply enjoyable. I also thank Dr. Oliver Lum and Dr. Rui Zhang for their detailed feedback on my work.

I was extremely fortunate throughout my studies to seek the advice and insights of Dr. Golden’s recent former students, especially Dr. Stefan Poikonen and Dr. Debdatta Sinha Roy.

Both a few years ahead of me in their professional careers, they were generous to provide their invaluable advice on mathematical methods, computational implementations, and research writing.

I also thank the faculty and staff comprising the Applied Math and Scientific Computing (AMSC) Program at the University of Maryland for their support during this journey, especially the AMSC coordinator, Jessica Sadler, for her impeccable responsiveness and communication.

I thank my parents and my sister for their constant and unconditional love and encouragement during my graduate school career. I thank my friend, Ryan, for stimulating lunchtime conversations during his time at University of Maryland. I must finally thank my girlfriend for her role as an uncompromising and limitless source of strength and perspective. I thank her for her patience during my countless late nights at the computer, sudden dashes towards pencil and paper, and, most heroically, impassioned impromptu seminars on scattered mathematical topics.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

B&P	Branch and Price
CG	Column Generation
EOL	end-of-line
eVTOL	Electric Vertical Takeoff and Landing (aircraft)
FTL	Full Truckload
GMM	Gaussian Mixture Model
LB	Lower Bound
LIFO	Last-In, First-Out
LKH	Lin-Kernighan Heuristic
LMP	Linearized Master Problem
LMSL	Last-Minute Service Location
LTL	Less-Than-Truckload
MIP	Mixed-Integer Program
MP	Master Problem
PSA	Parallel Savings Algorithm
RCESPP	Resource-Constrained Elementary Shortest Path Problem
RLMP	Reduced Linearized Master Problem
RVRP	Rendezvous Vehicle Routing Problem
STL	Shared Truckload
STLP	Shared Truckload Problem
TSP	Traveling Salesman Problem
UAM	Urban Air Mobility
UPS	United Parcel Service
USPS	United States Postal Service
VRP	Vehicle Routing Problem
VRPMS	Vehicle Routing Problem with Multiple Synchronization Constraints
VTOL	Vertical Takeoff and Landing (aircraft)

Chapter 1: Introduction

Since its introduction in 1959 [1], the Vehicle Routing Problem (VRP) has been among the most popular examples of the utility of mathematical programming, as well as one of its most well-studied applications. In its most basic form, the VRP seeks the minimum-distance set of routes for a fleet of vehicles such that they collectively serve a given set of customers. As a generalization of the Traveling Salesman Problem, the VRP is NP-Hard [2]. Nonetheless, its immediate applicability to a variety of important settings has led to its continued attention throughout the past several decades. In response to real-world applications, a wealth of variants have emerged. Such variations distinguish themselves from the basic VRP through new constraints (e.g., the VRP with time windows [3]), objectives (e.g., completion time rather than distance [4]), and/or assumptions about the availability of information and how/when decisions can be made (e.g., the Dynamic VRP [5]). Periodic efforts to taxonomize such variants (e.g., [6], [7], [8], [5], [9]) illustrate this plurality. Alongside this trend, a bifurcation in the nature of papers has emerged. On one end of the spectrum are endeavors to advance the state of the art on the more basic, yet nonetheless difficult, variants using exact methods (i.e., methods providing optimal solutions). On the other end are efforts to fully represent practical routing problems, leading to complex formulations that cannot be solved to optimality, and are instead solved using heuristic algorithms. The reliance on heuristic approaches is quite reasonable; in several applications, the problem sizes likely to be encountered exceed the limits of even the best exact approaches. On the other hand, exact methods allow researchers to characterize the nature of optimal solutions, which can inform the development of heuristics as well as provide a benchmark against which to test them. Furthermore, exact methods facilitate computational sensitivity analyses to problem parameters.

This dissertation seeks to straddle this gap through the introduction of three formulations representative of realistic problems solved using heuristic *as well as* exact methods. This is accomplished by the application of modern column generation (CG) approaches [10], an effective paradigm for routing problems that allows for a broad diversity of problem-specific constraints. Using CG methods on our problems, we are able to find optimal solutions to instances of non-trivial size. With access to such solutions, were are able to make qualitative observations about the nature of the best solutions (which can inform the development of heuristics), as well as provide quantitative assessments of heuristic methods (by comparing solution quality).

Each of the applications considered in this dissertation falls within the scope of the "Smart Cities" movement [11], a push towards more efficient urban areas in response to the numerous challenges arising from dramatic population growth in such regions. Indeed, the United Nations projects 68% of the world population will be urban by 2050 [12], which will intensify existing economic, social, and environmental issues in cities. The envisaged smart city will achieve its efficiency through integration of information and communication technology (ICT) and Internet of Things (IoT) technology into decision making, facilitating real-time responses that remove inefficiencies at a system-level (e.g., traffic lights changing in response to current traffic patterns). As transportation constitutes a vital component of urban life, models for scalable and efficient conveyance of people and goods plays a central role in the planning of Smart Cities. We consider applications, consistent with this movement, representing three distinct domains within transportation: freight routing, last-mile parcel delivery, and on-demand passenger transport. Each of the corresponding problems were developed under the advisement of partner companies invested in the corresponding space.

Furthermore, each of the problems is built around a novel idea with the potential to disrupt the status quo in their respective domains, and each anticipates a trend towards the Smart City paradigm.

In Chapter 2, we consider a routing problem emerging in a shared truckload (STL) freight service context, which is an alternative to full truckload (FTL) and hub-based, less-than-truckload (LTL) service. The STL model seeks to achieve the speed and reliability of FTL shipment and the cost-effectiveness of LTL shipment by pooling freight in trucks without consolidation at intermediate terminals; rather, the freight of a given shipper remains in the same truck from pickup to delivery. We take the perspective of a coordinating entity between shippers and carriers, who possesses a batch of requests from shippers and seeks to design truck routes that collectively serve all requests while minimizing the resulting carrier costs. It is desirable for a carrier to begin and terminate routes in areas of high demand, as trucks are then less likely to need to travel empty (i.e., deadhead) to begin a new route. Thus, we model costs as dependent on the demand levels at the original and terminal nodes and introduce the resulting shared truckload design problem. We then present a Branch and Price algorithm to arrive at optimal solutions for small (i.e., ≤ 25 customer) instances, as well as a column generation and parallel savings heuristic for finding near-optimal solutions on larger instances (up to 90 customers), the former within hours, and the latter within a minute. The column generation heuristic produces solutions within 1% of optimality for the largest instances. The parallel savings heuristic produces solutions within 3%

of the column generation solution, significantly outperforming a strong baseline approach, whose solutions are around 23% worse than optimal. We also carry out a sensitivity analysis of problem parameters, and evaluate the "price" of various constraints in the problem.

In Chapter 3, we consider a novel scheme for same-day delivery with a strong potential to reduce both transportation costs and carbon footprint as well as improve quality of service for consumers. A delivery company has two distinct fleets: trucks devoted to scheduled deliveries leaving on their fixed delivery routes early in the day, and shuttles leaving later in the day carrying same-day delivery packages. By allowing shuttles to intercept trucks and hand off packages for truck delivery, it may be possible to leverage the unfinished portion of truck routes to shorten the delivery routes of the shuttles. We refer to this as the rendezvous vehicle routing problem. We present a mathematical formulation of the problem, as well as a column generation algorithm that can quickly find optimal solutions for instances with up to 200 customer stops. We also develop and demonstrate the effectiveness of a specialized heuristic for use in larger instances with up to 1,000 customer stops. Our computational study validates the efficacy of truck-shuttle synchronization in this scheme, demonstrating savings of up to 77% in the test instances.

In Chapter 4, we consider a passenger transportation problem inspired by the Urban Air Mobility (UAM) movement. Over the next two decades, UAM Systems are anticipated to revolutionize the mass transportation industry. As envisioned presently, these systems will consist of electric vertical take-off and landing aircraft (eVTOLs) that operate from specially designed ports dispersed throughout a city. We consider the network logistics associated with the operation of a UAM system in its early phases, and focus on a problem of routing and scheduling eVTOLs to maximize passenger throughput. Key challenges for providers are the temporal nature of the demand, time windows for customers, and battery management constraints of the eVTOLs. We develop a three-index, arc-based formulation, routing eVTOLs over a time-expanded network. Due to the computational limitations of the arc-based formulation, we develop an alternate path-based formulation, and design a corresponding column generation procedure, identifying charge-feasible routes by way of a resource-constrained shortest path problem. The path-based approach is computationally robust, and can be applied in a heuristic manner by (i) sparsifying the time-expanded network, (ii) limiting column generation to the root node in a branch-and-bound scheme, and (iii) applying early termination criteria in the column generation procedure. Our computational experiments on a large set of test instances indicate that the path-based approach identifies high-quality solutions for large instances. We conduct a case study using Washington DC taxi data to demonstrate the viability of the column generation based heuristic.

Chapter 2: A Shared Truckload Service Problem with Origin and Terminal Point-Dependent Fares

2.1 Introduction

The freight industry represents a vital segment of the economy, and, to balance profitability and market demands, it requires economically efficient routes that achieve a high quality of service. The rise of e-commerce has been accompanied by an increased demand for lower-cost routes and shorter delivery windows [13]. Due to globalization of supply chains in the manufacturing and retail industries, there is a growing need to move relatively small loads that do not occupy a full truckload but are larger than what could be delivered by a parcel service (e.g., FedEx Ground, UPS) [14].

In response, the less-than-truckload (LTL) freight model pools together loads ranging from 150 to 15,000 lbs [15] from multiple shippers. Compared to the yield from transporting a full load associated with a single customer, the LTL model leads to significant savings for the shipper, which only needs to pay for a portion of the truck space, and greater revenue for the carrier. LTL providers employ a hub-and-spoke model, called the *line-haul* network, to facilitate freight consolidation and increase load factors (i.e., trailer utilization). The spokes of the network are regional *end-of-line* (EOL) terminals, which can serve only as origin or destination terminals, and the hubs are *breakbulk* terminals, which can serve as transfer points as well as origin or destination terminals. The load of a given shipper is first brought to the nearby terminal, where it is sorted and consolidated with other freight. The shipment then travels through the line-haul network towards the terminal near its destination, to which it is then finally delivered. The shipment may pass through multiple breakbulk terminals. Details on the design and operation of LTL networks are presented by Erera et al. [16].

The principal drawbacks of the LTL model compared to the full truckload (FTL) model are the increase in shipping time and risk of damage arising from the hub-and-spoke paradigm. In particular, shipments in terminals are typically sorted in a first-in, first-out (FIFO) manner, meaning entry into a terminal may correspond to entering a lengthy queue. Furthermore, freight must be unloaded and reloaded for consolidation at breakbulk terminals, leading to a significant risk of damage. For these reasons, LTL companies aim to allow no more than two handlings of freight at intermediate breakbulk terminals [17]. In contrast, an FTL service will deliver freight directly between the origin and destination specified by the shipper, which allows for fast delivery, reliable delivery time estimates, and a minimal risk of damage. Thus, a shipper must weigh the cost savings of LTL against the increase in shipping time and possibility of damaged freight. However, an alternative freight model has emerged which seeks to achieve the cost savings associated with shipment pooling while avoiding the drawbacks arising from the hub-and-spoke design. Rather than consolidating freight at breakbulk terminals, in the shared truckload (STL) model, the same truck will make the pickup and delivery for a given customer, traveling along a route consisting of a sequence of multiple pickups and deliveries. The risk of damage is minimized as freight is only loaded and unloaded once for each shipment, and the time from pickup to delivery is comparable to that of an FTL shipment. Furthermore, by avoiding trips to and from breakbulk terminals, the STL model can significantly reduce the carbon emissions generated.

In bypassing the hub-and-spoke model, which, given its structure, brings the benefit of simplifying route design, the success of the STL model requires carefully coordinated routes that can simultaneously meet service expectations and avoid under-utilization of the trailers. In this work, we take the perspective of a purchasing manager responsible for this coordination, which interfaces with both shippers (customers) and carriers. The shippers provide requests consisting of origin/destination pairs, descriptions of the freight (e.g., size, weight, freight class), and pickup dates. With this information, the manager designs routes that accommodate these requests and submits the routes for bidding among the carriers, seeking to pay as little as possible.

To model our problem, we assume the existence of a function that returns the cost of a route submitted to bidding among the carriers. In reality, carriers determine rates based on several parameters, and estimations of market rates require historical shipment data from a variety of carriers [14]. We assume that this has been done and that the historical data can be adequately fit by a known function of a particular form. Specifically, we assume this function is a linear combination of two quantities: the duration of the route and the number of stops. The weight of the duration (i.e., the per-hour fare) is dependent on the original and terminal locations. This is because it is in the interest of a carrier for a route to begin and end near a depot or near the terminals of other routes, as the trucks then do not need to travel great distances empty (i.e., deadhead) in order to be utilized again. We refer to such regions as high-demand regions, where demand denotes the attractiveness of the location for a carrier to begin or end a route. This captures the common use of industry standards called "tariffs," which price freight based on origin-destination ZIP codes (as well as freight class and weight). The per-mile rates incentivize routes starting and ending in high-demand locations. Thus, it may be advantageous for the purchasing manager to submit a route that ends in a high-demand area but takes more time than a route that serves the same customers and terminates in a low-demand area. Furthermore, the cost of a route increases with each additional stop added, with the increase referred to as the stop cost.

There are a number of constraints the coordinator must also consider. Precedence of pickup to delivery and truck capacity constraints must be respected. As STL providers adopt a one-touch policy, promising minimal risk of damage by allowing only a single loading/unloading for each shipment, shippers must be served in a last in, first out (LIFO) manner. We note that it may be possible to consider trucks capable of side-loading or loading shipments such that they can be unloaded arbitrarily (e.g., loading the left side and right side of a trailer independently). However, given the relative infrequency of these options, we do not consider them in this work. Furthermore, routes developed by the coordinator must not sacrifice the efficiency of STL shipping from the perspective of the customer. That is, the time from pickup to delivery should be comparable to the FTL alternative (i.e., the direct path). The size of the pools must also be limited. That is, the number of customers served by a given route cannot exceed a given value. According to our industry contact, as pooling at scale is a relatively novel concept in the industry, carriers still perceive extra time and distance as an inconvenience that they must be compensated for. While this can be captured by introducing a stop cost for each additional stop on a route, our industry partner has observed a breaking point in the number of stops such that carriers are unwilling to haul such a load at any price. In particular, a pool of more than four shipments is unlikely to be accepted. Finally, trucks are not permitted to be empty for any leg of the trip. While allowing such deadheads could lead to more cost-effective routes, our industry partner explains that such routes are not likely to be accepted by carriers.

In Section 2.2, we conduct a literature review of related problems in this space. In Section 2.3, we explicitly describe the assumptions, objectives, and constraints in our problem. In Section 2.4, we present a branch-and-price model to identify optimal or near-optimal solutions to small instances of the problem. In Section 2.5.2, we present our parallel savings algorithm to solve practically-sized instances. In Section 2.6, we conduct a computational study of our approaches, including a description of a test bed of instances we developed for the problem, and a sensitivity analysis of problem parameters and the LIFO constraints.

2.2 Literature Review

The STL freight model is a fairly new approach to freight shipping, and, as such, we are unable to identify any work directly addressing the domain. The LTL model, on the other hand, has existed for decades, and an abundance of research has focused on solving corresponding problems. While there is a significant difference in the organizational structure of the two strategies, there are several commonalities between existing work on LTL problems and our problem. In particular, many papers take the perspective of a coordinating entity between shippers and carriers, aiming to minimize carrier costs subject to satisfaction of service constraints. Furthermore, both freight models require intelligent routing *and* pooling methodologies to be successful. In both environments, ensuring high trailer utilization is an intuitive goal. Finally, as STL can be seen as an alternative to LTL, problem-specific parameters such as distances, load sizes, and length of planning horizon are similar. As the LTL industry is highly fragmented, modeling route costs is difficult. Similarly, we only expect our model of route costs to serve as a prediction. A regression model for costing LTL routes, synthesizing quantitative data and qualitative insights from practitioners, is presented by Özkaya et al. [14].

A survey of the various aspects of LTL routing, and contemporary freight more broadly, is presented by Crainic [18]. The author introduces a taxonomy of planning levels for transportation systems. Our work, similar to much of the research in LTL, falls within the *tactical* (medium-term) level, as we seek to design truck routes and shipment pools that minimize costs at a system level. The state-of-the-art structure in LTL planning is the *load plan*, which identifies a single sequence of terminals for each origin-destination pair in the network. Given a load plan, LTL providers create a *planned flow*, which is the actual sequence of truck routes along which freight travels. In addition, an alternative sequence for each origin-destination pair may be included in the load plan, which allows for greater flexibility for real-time adjustments of the planned flow [19]. Powell and Sheffi [20] present a two-phase approach to generate the load plan, seeking to minimize the number of trailers utilized. Katamaya and Yurimoto [21] formulate the load-planning problem as an arc-based IP, which is solved using a Lagrangian relaxation method. A slope-scaling heuristic to solve a tree-based formulation of the problem is presented by Jarrah et al. [17]. Erera et al. [22] present an integer program-assisted neighborhood search algorithm. An Ant Colony optimization approach is presented by Barcos et al. [23]. A Steiner-forest based heuristic to solve for routing LTL freight through a time-expanded network is presented in Tamvada et al. [24].

The STL routing problem can be viewed as a variant of the Dial-a-Ride Problem (DARP) [25], itself a generalization of the Pickup and Delivery Problem (PDP) [26], both well-studied. Similar to our problem, the PDP seeks the design of routes that transport customers (or, their goods) from origins to destinations, where customers can occupy a vehicle simultaneously. Like our problem, constraints are placed on precedence of pickup to delivery, the capacity of vehicles, and, in the case of the DARP, the elapsed time from pickup to delivery. However, the objective of the STL routing problem is quite different from popular PDP objectives, such as the minimization of travel time, travel distance, or number of vehicles required. In particular, the objective in our work depends greatly on the starting and ending nodes of selected routes. To our knowledge, no existing application of the PDP or the DARP costs routes in such a way. Rather, most PDP formulations introduce a depot, representing the common starting and ending location for all routes.

Given their practical relevance to the logistics industry, there has been effort within the vehicle routing research community to capture LIFO constraints. Solution methodologies for PDPs with LIFO loading include exact [27, 28, 29, 30] and heuristic [31, 32, 33, 34] approaches. Successful heuristics have been based on variable neighborhood search. However, to our knowledge, there is no work on incorporating LIFO constraints along with the other constraints outlined in Section 2.1, such as limiting the duration from pickup to delivery for each customer.

Our heuristic makes use of a parallel savings algorithm (PSA) for merging routes. Similar algorithms for variants of the Vehicle Routing Problem have been proposed. Desrochers and Verhoog [35] and Altinkemer and Gavish [36] demonstrated its use for solving the Capacitated VRP. Gajpal and Abad [37] use a similar PSA for a VRP with simultaneous pickup and delivery. However, to our knowledge, the PSA has not been used in conjunction with a set partitioning formulation, the corresponding matheuristic presented in our work.

2.3 Problem Description

We consider a set of n customers, where each customer is associated with a pickup node and a delivery node. Let $n = |C|$, $P = \{1, \ldots, n\}$ be the set of pickup locations and $D = \{n + 1, \ldots, 2n\}$ be the set of delivery locations. We assume the delivery location for the load picked up at $i \in P$ is $i + n \in D$. Let $V = P \cup D$. Each $i \in V$ is associated with a demand level, $d_i \in \mathbb{R}^+$. Let A be the set of arcs (i, j) between every pair of nodes $i, j \in V, i \neq j$. Let c_{ij} be the travel time to traverse arc (i, j). We assume $\mathbf{c}$ satisfies the triangle inequality. For each $i \in P$, let $q_i > 0$ be the load of the item that must be picked up at i and delivered at $i + n$. Let $Q \in \mathbb{R}$ be the uniform truck capacity.

A truck route, r, associated with a sequence of nodes $\{s_r = r_0, r_1, \ldots, r_{|r|-1} = t_r\}$, where $r_k \in V$ for $k \in \{0, \ldots, |r| - 1\}$, must visit both the pickup and delivery node for each customer it visits, visiting the pickup node first, but it can begin at any pickup node and end at any delivery node. That is, we assume trucks are ubiquitous. In practice, carriers have an initial deadhead segment to get to their first stop. However, our industry partner also ignores this leg, with the justification that the carrier market is quite dense. Thus, the purchasing manager should be able to find a carrier close enough to the first stop so that the initial deadheading leg can be ignored. Respecting the one-touch policy, the pickup and delivery sequence must satisfy LIFO constraints. The time traveled along the route is given by $c_r = \sum_{i=0}^{|r|-2} c_{r_i, r_{i+1}}$. There is a per-unit time cost associated with a route originating at node i and terminating at node j, $F(i, j) \in \mathbb{R}^+$. This value is dependent only on the demand levels at i and j. That is, there is a function $g : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ such that $F(i, j) = g(d_i, d_j)$ for all $i, j \in N$. Furthermore, for each route r, there is a stop cost, $e_r \in \mathbb{R}^+$, which is linear with respect to each customer, after the first, added to the route.

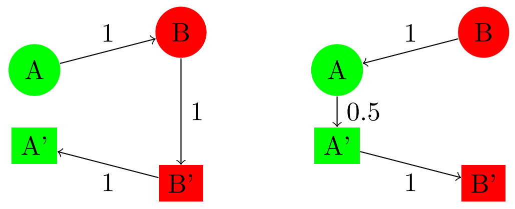


Figure 2.1: Comparison of two STL routes.

That is, $e_r = E \max\{(|r| - 2)/2, 0\}$, where $E \in \mathbb{R}^+$ is the per-additional customer stop cost. Trucks cannot travel empty at any point (i.e., deadheads are not permitted). For each customer, the elapsed time from pickup to delivery cannot exceed a given factor, $\lambda \in \mathbb{R}$, of the time to go directly from the pickup location to the delivery location. A given route cannot serve more than $\tau \in \mathbb{Z}$ customers. At any point, the load on a truck cannot exceed Q. We compare two possible routes in Figure 2.1 which serve two customers. A and A' denote the pickup and delivery location for the first customer, and B and B' those for the second. Each arc is labeled by its associated travel time. Both routes satisfy LIFO constraints and never deadhead. The first route has a total travel time of 3, and the second, 2.5. However, the pickup and delivery for the first customer are both high demand (represented by green nodes), whereas those for the second are low demand (represented by red nodes). Correspondingly, the first route will have a lower fare. If that fare is sufficiently lower than the fare of the second route (in this case, below a factor of $5/6$), the first route is preferable. We also note that the first route is only feasible if $\lambda \geq 6$.

We seek the set of truck routes, R, which serves all customers exactly once and minimizes:

$$\sum_{r \in R} \left[F(s_r, t_r) c_r + e_r \right].$$

We call this the shared truckload problem (STLP). The problem data is summarized in Table 2.1. It is possible to model the STLP using an integer linear formulation by defining a

Name	Description
$n \in \mathbb{Z}$	Number of customers
$P = \{1, \ldots, n\}$	Set of pickup locations
$D = \{n+1, \ldots, 2n\}$	Set of delivery locations
$V = P \cup D$	Set of all locations
$d_i \in \mathbb{R}^+$	Demand level at location $i \in V$
$q_i \in \mathbb{R}^+$	Load associated with pickup $i \in P$
$c_{ij} \in \mathbb{R}^+$	Travel time from $i \in V$ to $j \in V$
$F(i,j) \in \mathbb{R}^+$	Per-unit time cost for starting at node $i \in P$ and ending at node $j \in D$
$E \in \mathbb{R}^+$	Per-additional customer stop cost
$Q \in \mathbb{R}^+$	Uniform truck capacity
$\tau \in \mathbb{Z}^+$	Maximum number of customers a single truck can serve
$\lambda \in \mathbb{R}^+$	Maximum delay factor

Table 2.1: Problem Parameters for the STLP

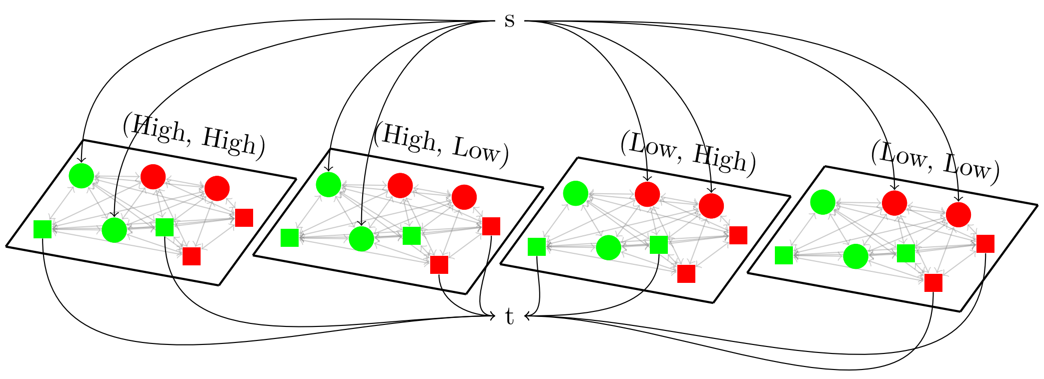


Figure 2.2: Visualization of the fare-expanded network.

fare-expanded network, which we now construct.

Let $F = \{f \in \mathbb{R} \mid \exists i \in P \text{ s.t.} f = F(i, i+n)\}$ be the (finite) set of fares. Note that due to LIFO constraints, as well as the prohibition of deadheading, all feasible routes starting at $i \in P$ must terminate at node $i + n \in D$. For $i \in P$, we say i (respectively, $i + n$) is an *entry node* (respectively, *exit node*) for fare $f \in F$, if a route starting at i has fare f.

We define the fare-expanded nodes $M = V \times F$, where a member of M is a location-fare pair (i.e., the fare-expanded node $(v, f) \in M$ corresponds to location $v \in V$ and fare $f \in F$).

Let s and t be fictional start and end nodes, respectively. Let B be the set of arcs between nodes $(v,f),(v',f') \in M$ such that $f = f'$, as well as arcs of the form $(s,(i,f))$, where $(i,f) \in M$ and i is an entry node for fare f. Also include in B arcs of the form $((i,f),t)$, where $(i,f) \in M$ and i is an exit node for fare f. Since no arcs in B connect members of M associated with different fares, and since s and t are not connected, we may unambiguously denote a member of B by (i,j,f), where, if $i = s$, f is the fare associated with the head node, and if $j = t$, f is the fare associated with the tail node. An example of such a *fare-expanded* network, $(M \cup \{s,t\}, B)$, with four possible fares, is presented in Figure 2.2. The travel network is copied for each fare. Circles correspond to pickup nodes, and squares correspond to delivery nodes. Green nodes correspond to high demand, and red nodes to low demand. Arcs extend from the fictional starting node, s, towards the entry nodes in each of the sub-networks. Arcs extend from the exit nodes in each of the sub-networks towards the fictional end node, t. Each sub-network is fully connected. Note that arc costs may be defined in this fare-expanded network such that the cost of a route is the sum of the arc costs. A path from s to t through the left-most network will correspond to a route starting and ending at a high-demand location, which will have a low per-mile fare. Define the cost of travel along arc (i,j,f):

$$c_{ij}^{f} = \begin{cases} 0 & i = s \\ 0 & j = t \\ f \cdot c_{ij} + E & i \neq s, j \in P \\ f \cdot c_{ij} & j \in D, \end{cases}$$

where we include the stop cost, E (associated with visiting an additional customer) by associating it with the travel to their pickup node, provided the customer is not the first served by the truck (i.e., the node visited immediately before the customer's pickup node is not s).

Let the binary decision variable $x_{ij}^f \in \{0, 1\}$ for $(i, j, f) \in B$ indicate the use of arc (i, j), traveling with fare f. Let the binary decision variable $u_{ik} \in \{0, 1\}$ for $i \in V \cup \{s, t\}$, $k \in P$ equal 1 if the freight associated with pickup location k is in the trailer when a truck departs from location i. The u_{ik} variables are similar to those used by Benavent et al. [29] to include LIFO constraints in the multi-vehicle pickup and delivery problem. They are inspired by the Miller-Tucker-Zemlin subtour elimination constraints for the traveling salesman problem [38]. However, rather than representing cumulative loads, which grow monotonically along the routes, the variables represent loads of k distinct commodities that are picked up and dropped off. We define the coefficients β_{ik} for $i \in V \cup \{s, t\}$ and $k \in P$ with

$$\beta_{ik} = \begin{cases} 1 & i = k \\ -1 & i = k + n \\ 0 & \text{otherwise,} \end{cases}$$

noting that β_{ik} indicates that item k can only be picked up from node k and can only be delivered at node $k + n$. Let the coefficients γ_i for $i \in V \cup \{s, t\}$ equal 1 if and only if $i \in P$. Let the continuous decision variable α_i for $i \in V \cup \{s, t\}$ denote the time of departure from node i. Let the continuous decision variable v_i for $i \in V \cup \{s, t\}$ denote the number of pickups made by the truck upon departure from node i. The following program models the STLP:

$$\text{Minimize:} \sum_{(i,j,f) \in B} c_{ij}^f x_{ij}^f \tag{2.1}$$

$$\begin{aligned}
\text{Subject to:} \quad \sum_{\{f\in F|(i,j,f)\in B\}} x_{ij}^f &= y_{ij} \qquad \forall i,j \in V\cup\{s,t\} && (2.2)\\
\sum_{\{j\in V|(i,j)\in A\}} y_{ij} &= 1 \qquad \forall i \in V && (2.3)\\
\sum_{\{i\in V|(i,j)\in A\}} y_{ij} &= 1 \qquad \forall j \in V && (2.4)\\
\sum_{\{j\in V|(i,j,f)\in B\}} x_{ij}^f &= \sum_{\{j\in V|(j,i,f)\in B\}} x_{ji}^f \qquad \forall i \in V, f\in F && (2.5)\\
\alpha_i + c_{ij} - M_1(1-y_{ij}) &\leq \alpha_j \qquad \forall i \in V, j \in V && (2.6)\\
(\alpha_{n+i}-\alpha_i) &\leq \lambda c_{i,n+1} \qquad \forall i \in P && (2.7)\\
v_i + \gamma_j - M_2(1-y_{ij}) &\leq v_j \qquad \forall i \in V, j\in V && (2.8)\\
v_i &\leq \tau \qquad \forall i \in P && (2.9)\\
u_{sk} &= 0 \qquad \forall k \in P && (2.10)\\
u_{tk} &= 0 \qquad \forall k \in P && (2.11)\\
u_{kk} &= 1 \qquad \forall k \in P && (2.12)\\
u_{n+k,k} &= 0 \qquad \forall k \in P && (2.13)\\
u_{ik} &= u_{i+n,k} \qquad \forall i \in P, k\in P, i\neq k && (2.14)\\
\sum_{k\in P} q_k u_{ik} &\leq Q \qquad \forall i \in P && (2.15)\\
\sum_{k\in P} u_{ik} &\geq 1 - y_{it} \qquad \forall i \in D && (2.16)\\
u_{ik} - u_{jk} + y_{ij} + (1-\beta_{ik}-\beta_{jk})y_{ji} &\leq 1-\beta_{jk} \qquad \forall i,j \in V\cup\{s,t\}, k\in P && (2.17)\\
u_{ik} - \sum_{j\neq i}\beta_{jk}y_{ji} &\geq \beta_{ik} \qquad \forall i \in V, k \in P && (2.18)\\
x_{ij}^k &\in \{0,1\} \qquad \forall (i,j,k)\in B && (2.19)\\
u_{ik} &\in \{0,1\} \qquad \forall i \in V, k\in P && (2.20)\\
\alpha_i &\in \mathbb{R} \qquad \forall i \in V\cup\{s,t\} && (2.21)
\end{aligned}$$

$$v_i \in \mathbb{R} \qquad \forall i \in V \cup \{s,t\}. \tag{2.22}$$

The objective 2.1 is to minimize the sum of the arc costs. Constraints 2.2 define the auxiliary variable y_{ij}, which is the aggregation of the x^f_{ij} variables over f. This aggregation serves to simplify the remaining constraints. Constraints 2.3 and 2.4 require each pickup and delivery location is visited exactly once. Constraints 2.5 require continuity of flow within each sub-network. Constraints 2.6 capture the propagation of departure times from each node, and constraints 2.7 ensure that the elapsed time from pickup to delivery for each package is within a factor of λ of the time associated with the direct path. Here, M_1 is a sufficiently large number, and can be set to $(2\tau - 1)\max_{i,j} c_{ij}$. Constraints 2.8 capture the propagation of the number of customers served by routes, and constraints 2.9 limit these values to τ. Here, M_2 is a sufficiently large number, and can be set to τ. Constraints 2.10 and 2.11 require trucks to be empty at the beginning and end of their routes. Constraints 2.12 and 2.13 require each load to be present in the truck upon departure from the pickup node and absent from the truck upon departure from the delivery node. Constraints 2.14 ensure the LIFO policy is respected, constraints 2.15 limit the load on a trailer at any point to Q, and constraints 2.16 prevent trucks from traveling empty within their route. Constraints 2.17 and 2.18 ensure correct propagation of truck loads. Ignoring the fourth term on the left-hand side (which is used for strengthening the formulation), constraints 2.17 can be read as $y_{ij} = 1 \Rightarrow u_{ik} + \beta_{jk} \leq u_{jk}$. That is, given that a truck travels from i directly to j, the presence of package k in the trailer upon departure from j depends on whether it was present upon departure from i and on whether j is its pickup or delivery location. Similarly, constraints 2.18 can be read as $y_{i'i} = 1 \Rightarrow \beta_{i'k} + \beta_{ik} \leq u_{ik}$. Proofs for the validity of constraints 2.17 and 2.18 follow the same lines as those given by Benavent et al. [29], and are presented in Appendix A.1. Constraints 2.19 and 2.20 ensure integrality of the x_{ij}^k and u_{ik} variables, and constraints 2.21 and 2.22 permit continuous arrival times and cumulative pickups.

2.4 Branch-and-Price

The formulation presented in Section 2.3 is only suitable for modestly-sized instances. To handle larger instances, we present a branch-and-price algorithm to solve the STLP. The STLP is modeled as a set-partitioning formulation with an exponential number of variables (each representing a feasible route). This formulation is solved using column generation. Initially, possible routes are restricted to a small set. After solving the linear relaxation of this restricted set partitioning problem, we may use the generated dual variables to identify routes to add to the set (i.e., we generate columns). This step is called the "pricing problem." We describe how we solve the pricing problem in Section 2.4.1. We iterate this process (solving the linearized set partitioning and pricing problems) until the pricing problem is unable to identify routes to be added. At this point, we have confirmation that the incumbent solution to the linear set-partitioning problem is optimal. Because the resulting solution may be fractional, this entire process is embedded within a branch-and-bound tree. The manner in which we branch is described in Section 2.4.2.

We now explicitly describe our column generation procedure. Let Ω be the set of feasible truck routes, let the coefficient $a_\omega^c \in \{0, 1\}$ denote whether the customer associated with pickup $i \in P$ is served by route $\omega \in \Omega$, and let

$$z_\omega = F(s_\omega, t_\omega)c_\omega + q_\omega$$

be the cost of route ω. Finally, let the decision variable $y_\omega \in \{0, 1\}$ denote whether truck path $\omega \in \Omega$ is used in the solution. The problem is modeled by the following set partitioning formulation, denoted as MP (master problem):

$$\begin{aligned} \min \sum_{\omega \in \Omega} z_\omega y_\omega & & & \\ \sum_{\omega \in \Omega} a^i_\omega y_\omega &= 1 & \forall i \in P & \\ y_\omega &\in \{0,1\} & \forall \omega \in \Omega. & \end{aligned} \tag{2.23}$$

We can solve the linear relaxation of MP via column generation. We denote the linearization of MP as LMP (linearized master problem) and denote the restriction of LMP to a small set of routes $\Omega_0 \subset \Omega$ by RLMP (reduced linearized master problem). We assume Ω_0 is such that the RLMP is feasible (perhaps constructed by generating the n truck routes that each serve a single customer). Let $\{u_i\}_{i \in P}$ be the dual variables for the set partitioning constraints 2.23 in the RLMP. After solving RLMP via the Simplex method, we search for dual-infeasible routes for LMP, i.e., routes ω satisfying

$$z_\omega - \sum_{\{i \in P | a^i_\omega = 1\}} u_i < 0.$$

The search for such routes can be expressed as a shortest path problem with side constraints through the fare-expanded network, with arc costs modified based on the dual variables u_i. The arc costs are updated so as to reward a value of u_i to a route for serving customer i. Specifically, for each arc $(i, j, f) \in B$, we set the modified cost:

$$\hat{c}^f_{ij} = \begin{cases} c^f_{ij} - u_i & j \in P \\ c^f_{ij} & j \notin P. \end{cases}$$

Defining arc-costs in this way ensures that $\hat{c}^f_{ij} + \hat{c}^f_{jk} \geq \hat{c}^f_{ik}$ if $j \in D$. This property, called the *delivery triangle inequality* by Ropke and Cordeau [39] in the context of the Pickup and Delivery Problem with Time Windows, assists in defining a dominance criterion in a labeling algorithm, which can be used to solve the pricing problem. The pricing problem can then be expressed:

$$\begin{aligned} \min \sum_{\{(i,j,f) \in B\}} & \hat{c}^f_{ij} x^f_{ij} \\ \text{Subject to: } \sum_{i \in P} \sum_{\{f \in F | (s,i,f) \in B\}} x^f_{si} &= 1 \\ \sum_{i \in D} \sum_{\{f \in F | (i,t,f) \in B\}} x^f_{it} &= 1 \\ \text{Equations 2.2 and 2.5-2.22}, & \end{aligned}$$

where the first and second constraint force a path to start at s and end at t.

2.4.1 Labeling Algorithm

We employ a labeling algorithm to solve the pricing problem, which is centered around a data structure called a label. A label corresponds to a partial path through the network, starting at s. A label is extended by considering all feasible extensions of the associated path by a single node, which generates new labels. Beginning with an initial label associated with a path of length 0 starting at s, labels are iteratively extended (yielding new labels) until no feasible extensions exist. In the interest of speeding up the labeling algorithm, we delete from the network certain arcs that cannot be part of a path of minimum cost. Specifically, we remove all arcs of the form $(i+n, i, f)$ and all arcs of the form (i, j, f) where $i \in P$ and $j \in D \backslash \{i+n\}$. That is, we delete arcs connecting the customer departure locations to pickup locations, as well as all arcs connecting a customer's pickup location with the delivery locations for the other customers (an impossible connection given the LIFO constraints). We also remove arcs of the form (i, j, f) where $i, j \in P$ and either $q_i + q_j > Q$ or $c_{ij} + c_{j,j+n} + c_{j+n,i+n} > \lambda d_{ij}$, as paths including such arcs violate capacity or time limit constraints, respectively.

In our algorithm, a label is represented $\mathcal{L} = (i^l, z^l, q^l, \mathcal{V}^l, \mathcal{T}^l, \mathcal{C}^l)$. For label $\mathcal{L}$, i^l is the final node on the associated path, z^l is the accumulated cost along the path, and q^l is the current load on the truck. $\mathcal{V}^l \subseteq M$ is the set of nodes that have been visited along the path. $\mathcal{T}^l = [T_1^l, \ldots, T_n^l]^T$ is a vector representing the remaining time available to serve each customer (i.e., the remaining time to serve customer i, given the partial path associated with the label, is T_i^l). If the pickup location customer i has not been visited, or both the pickup and delivery locations have been visited, $T_i^l = \infty$. Finally, $\mathcal{C}^l$ is a set of indices associated with customers whose loads are currently in the truck. As customers are served in a LIFO manner, C^l is accessed as a stack. To recover the partial path associated with a label, each label also contains a pointer to its previous label. We define $q_{i+n} = -q_i$ for all $i \in P$, and $q_t = 0$.

The algorithm begins with the initial label $(s, 0, 0, \emptyset, \{\infty\}_{i \in P}, \emptyset)$, which is added to a set of untreated labels, U. While the set of untreated labels is not empty, a member is selected and is treated. When treating a label $\mathcal{L}$, we consider all feasible extensions of the associated path. Specifically, we consider all $(i, j, f) \in B$ such that $i = i^l$, $q^l + q_j \leq Q$, $j \notin V^l$, $T_k^l - c_{ij} > 0$ for all $k \in P$, and, if $j \in D$, $j - n$ is the top element of C^l. Given such an arc, we create a new label, $\mathcal{L}' = (i^{l'}, z^{l'}, q^{l'}, \mathcal{V}^{l'}, \mathcal{T}^{l'}, \mathcal{C}^{l'})$, where the components are defined as follows:

$$\begin{aligned}
i^{l'} &= j \\
z^{l'} &= z^l + \hat{c}^f_{ij} \\
q^{l'} &= q^l + q_j \\
V^{l'} &= V^l \cup \{j\} \\
T^{l'}_k &= \begin{cases} \lambda c_{k,k+n} & k = j, j \in P \\ \infty & k = j, j \in D \\ T^l_k - c_{ij} & k \neq j, T^l_k < \infty \\ \infty & k \neq j, T^l_k = \infty \end{cases} \\
\mathcal{C}^{l'} &= \begin{cases} \mathcal{C}^l \cup \{j\} & j \in P \\ \mathcal{C}^l \setminus \{j\} & j \in D. \end{cases}
\end{aligned}$$

The resulting label is added to U. Once a label is treated, it is added to T. The algorithm continues until U is empty. Then, T is searched for labels L such that $i^l = t$. Of those labels, the one with the lowest z^l corresponds to the minimum reduced cost path.

To prevent the labeling algorithm from exploring verifiably suboptimal paths, for each newly generated label l, a search is conducted among the previously generated labels for a label $l' \in U \cup T$ that dominates l. A label l' dominates another label l if the following conditions hold:

$$\begin{aligned}
i^{l'} &= i^l \\
z^{l'} &\leq z^l \\
O^{l'} &\subseteq O^l
\end{aligned}$$

$$T_k^{l'} \;\geq\; T_k^l \qquad \forall k \in O^{l'}$$

where $O^l = \{i \in \mathcal{V}^l \mid i \in P \text{ and } i+n \notin \mathcal{V}^l\}$ is the set of customers whose pickup has been visited along the path associated with l but whose delivery has not; that is, O^l is the set of *open customers* associated with l. We may observe that if l' dominates l, then l may be removed from consideration. We see that the path associated with l' has fewer obligated deliveries to make than that of l, and, because of the delivery-triangle inequality, l cannot get an advantage by making a delivery that l' cannot. Furthermore, by the final condition, any additional customers added to l can be added to l', since there is at least as much time to make all subsequent deliveries.

2.4.2 Branching

In the event of fractional solutions to the LMP, we must branch the solution space. Branching on the variables y_ω is undesirable, as it is nontrivial to adjust the pricing problem in a compatible way. It is, therefore, the standard approach in the routing literature to branch on the arcs in the underlying network. If the solution to the LMP is fractional, we use the following strategy. Let $\hat{\mathbf{y}}$ be the primal solution to the RLMP. For each arc $(i,j) \in A$, let $H_{ij} \subseteq \Omega$ be the set of routes that use the arc. Define the flow for the arc $\sum_{\omega \in H_{ij}} \hat{y}_\omega$. We branch on the arc with the flow closest to 0.5. In the event of ties, we branch on the tying arc of minimum lexicographic order. To prohibit the arc (i,j), we remove from B all arcs of the form (i,j,f). To enforce the arc, we remove all arcs of the form (i,k,f) where $k \neq j$ and (k,j,f) where $k \neq i$.

2.4.3 Extensions

We investigate acceleration strategies for the above branch-and-price algorithm. In particular, we consider solving the pricing problem in a lazy manner. Rather than solving the shortest path problem on the entire fare-expanded network, we initially solve it on the lowest-fare sub-network (e.g., the left-most sub-network in Figure 2.2). Failing to produce columns from that network, we then solve the pricing problem on the second lowest-fare sub-network. Failing to produce columns, we solve on the third lowest-fare sub-network, and so on. The rationale for this method is that "good routes" are more likely to be found by exploring the lower fare sub-networks, and time can be saved by avoiding exploration of the higher-fare sub-networks. However, in order to prove optimality, each sub-network must be evaluated. Our experiments have shown that, typically, the pricing algorithm is called at most twice when solving by column generation. The first time, a large number of routes are produced, and the second time, none. Thus, there is little benefit to lazily solving the pricing problem.

A heuristic version of the above described branch-and-price algorithm limits column generation to the root node of the branch-and-bound tree. This approach produces a lower bound on the true optimal solution. Meanwhile, the set partitioning problem, restricted to the columns generated at the root node, can provide a high quality feasible solution. Our experiments, presented in Section 2.6, demonstrate this heuristic is an effective solution methodology for the STLP. A further extension of the root node heuristic restricts the pricing problem to the lower-fare sub-networks. To accomplish this, arcs of the form (s, i, f) are removed from the network, where f is a higher fare. This prohibits the generation of routes with high fares, and speeds up the labeling algorithm. However, we have found that the time savings associated with this strategy do not justify the resulting deterioration of solution quality. In particular, for a 24 customer problem with nine fare subnetworks, removing the three highest fares led to drops in solution quality as large as 17%.

2.5 Heuristics

We now describe the heuristics we developed to solve larger instances of the STLP than can be solved by the above-described approaches in reasonable computing time. We describe a baseline heuristic in Section 2.5.1, and then a heuristic based on parallel savings in 2.5.2. We also describe a matheuristic extension of the parallel savings algorithm in 2.5.2.1.

2.5.1 Baseline Heuristic

We define a 1-route as a route serving a single customer. We call the solution consisting of all 1-routes the trivial solution. Note that in the trivial solution, each customer gets their own truck. The baseline heuristic starts from the pool of all 1-routes, R. The route with the maximum cost, r, is selected. For each 1-route in R, r', we identify the minimum-cost combination of r and r'. We denote the minimum-cost route among the feasible resulting combinations by r^c, and the associated 1-route by r^*. If the combination leads to overall savings, the constituent routes are removed from the pool and the combined route is added. Otherwise, r is added to a set of "final routes," F, and removed from R, and the process repeats until R is empty. Then, the routes in F form a feasible solution. The pseudocode for this heuristic is presented in Algorithm 1.

On a set of ten 500-customer instances, the baseline heuristic takes an average of 5.98 seconds with a sample standard deviation of 0.98 seconds. For the same instances, the average

Algorithm 1 Baseline Heuristic

```
1: function BASELINE
2:     $R \leftarrow$ all 1-routes
3:     $F \leftarrow \emptyset$
4:     while $R \neq \emptyset$ do
5:         $r \leftarrow$ GETMAXCOSTROUTE($R$)
6:         $r^c, r^* \leftarrow$ GETBESTCOMBINATION($r$, $R$)
7:         if COST($r^c$) $<$ COST($r$) + COST($r^*$) then
8:             $R \leftarrow R \backslash \{r, r^*\}$
9:             $R \leftarrow R \cup \{r^c\}$
10:        else
11:            $R \leftarrow R \backslash \{r\}$
12:            $F \leftarrow F \cup \{r\}$
13:        end if
14:    end while
15:    return $F$
16: end function
```

improvement on the trivial solution is 70.91%.

2.5.2 Parallel Savings Algorithm

We now describe a parallel savings algorithm we employ to find stronger solutions than those found by the baseline heuristic. Starting from some feasible solution (for example, the trivial solution), we construct a complete network. Each node in the network corresponds to a route in the solution, and the cost of each edge is the savings associated with merging together the associated routes (if it is possible to do so). Specifically, if the cost of a route r is z_r, and the lowest-cost merging of route a and route b has cost z_{ab}, then the cost of the edge connecting the nodes associated with route a and route b is $z_{ab} - z_a - z_b$. Note that it may not be feasible to merge a and b, in which case we set $z_{ab} = \infty$. If a solution has $|R|$ routes, then $\binom{|R|}{2}$ such values are computed. The minimum cost merging of route a and b, z_{ab}, can be computed or approximated. We note that given the LIFO constraints and the prohibition of deadheading, the number of feasible combinations of two routes is fairly modest by observing the following:

Theorem 1. *There are $(2n-2)!/(n-1)!$ routes serving a given group of n customers which satisfy LIFO constraints and avoid deadheading, but ignore temporal and capacity constraints.*

Proof. Consider a route serving n customers, satisfying LIFO constraints, with a given order of pickup locations visited. Observe that such a route can be uniquely associated with a monotonic lattice path along the edges of a $n \times n$ grid, starting at $(0,0)$ and ending at (n,n), which does not pass below the diagonal. A pickup is associated with moving upwards, and a delivery with moving right. The number of such paths is among the interpretations of the nth Catalan number, $C_n = \frac{1}{n+1}\binom{2n}{n}$ [40].

Observe that if the route never deadheads, the corresponding lattice path only meets the diagonal at the endpoints $(0,0)$ and (n,n). Note that the second position in the path must be $(0,1)$, and the penultimate position must be $(n-1,n)$. Thus, the number of paths only meeting the diagonal at $(0,0)$ and (n,n) is equal to the number of paths from $(0,1)$ to $(n-1,n)$ which do not pass below the diagonal line connecting $(0,1)$ to $(n-1,n)$, namely, C_{n-1}. Since we may serve the customers in $n!$ distinct sequences, the total number of routes is then $(n!)C_{n-1} = (2n-2)!/(n-1)!$. $\square$

Therefore, if τ is set to 4, as is the case in our instances, there are at most $(2 \cdot 4 - 2)!/(4-1)! = 120$ combinations that need to be considered, since we do not combine two routes if the total number of customers exceeds τ. Therefore, we may compute the exact savings in our implementation by enumerating all such combinations. Note that feasibility checks for such routes only need to verify satisfaction of temporal and capacity constraints.

Once the savings are computed for each edge, we identify the routes to combine by solving the maximum weight matching problem on the network. This is done in $\mathcal{O}(|V|^2|E|)$ time, where $|V|$ and $|E|$ are the number of vertices and edges in the network, respectively, using an open-source algorithm [41] based on Edmonds' "blossom" method [42]. This yields a set of pairs of routes whose combination improves the objective value. We construct a new solution by removing each matched route from the original solution and adding the combined routes. We then repeat, starting from the new solution, until there is no positive-weight matching discovered by the matching algorithm. Pseudocode for the heuristic is presented in Algorithm 2. Starting from an initial solution S, a complete network, G, is constructed, where the vertices, V, correspond to routes in the solution. For each edge, e, in the network, the savings associated with combining the associated routes is computed (`GetSavings(e)`). We then solve the maximum weight matching problem on the network to produce the matching m, a set of pairs $(v_1, v_2) \in V \times V$ such that the combination of the associated routes produces positive savings. We then produce a new solution, S', which modifies the original solution by replacing all routes that appear in the matching with the combined routes (`GetMerging(S,m)`). If the new solution does not lead to an improvement in the objective, we raise a flag to terminate. Otherwise, we set S to S' and repeat.

The parallel savings algorithm repeatedly coalesces routes until a local minimum is reached. The quality of the final solutions is sensitive to the routes in the input solutions. We embed the parallel savings algorithm in a multi-start procedure, hoping to identify multiple local minima. We produce M initial solutions using the following simple procedure. First, the set of all 2-routes is generated, where a 2-route is a route that serves exactly two customers and does so with minimum cost. Note that there are only two possibilities for such a route. The 2-routes are then sorted by their savings, where the savings of a 2-route is the cost of the associated 1-

Algorithm 2 Parallel Savings Algorithm

```
 1: function MERGING
 2:     $S \leftarrow$ initial solution
 3:     Flag $\leftarrow$ False
 4:     while Flag $=$ False do
 5:         $V \leftarrow \{$Routes in $S\}$
 6:         $E \leftarrow \{(v_1, v_2) \mid \forall v_1, v_2 \in V\}$
 7:         $G \leftarrow (V, E)$
 8:         for $e \in E$ do
 9:             weight$(e) \leftarrow$ GETSAVINGS$(e)$
10:         end for
11:         $m \leftarrow$ MAXIMUMWEIGHTMATCHING$(G)$
12:         $S' \leftarrow$ GETMERGING$(S, m)$
13:         if cost$(S) =$ cost$(S')$ then
14:             Flag $\leftarrow$ True
15:         end if
16:         $S \leftarrow S'$
17:     end while
18:     return $S$
19: end function
```

routes minus the cost of the 2-route. Then, for each of the first M 2-routes in the sorted list, a solution is generated by replacing the 1-routes in the trivial solution with the 2-route. Note that the savings of a 2-route may be negative, meaning the perturbed solution can be *worse* than the trivial solution. Nevertheless, we include such routes, for the following reason. Suppose a route serving customers A, B, and C appears in the optimal solution, but the savings associated with each route serving a pair of the three is negative. Then, it is impossible for the PSA to find the optimal route. While the solutions produced by this procedure are only small perturbations of the trivial solution, experiments have demonstrated this simple diversification procedure leads to significant improvement in output solutions. Furthermore, we found that more elaborate initialization schemes tended to perform no better than the one we have described. In our work, we generate $M = 16$ initial solutions in this manner, and also include the baseline solution generated using the process described in Section 2.5.1.

2.5.2.1 PSA as Column Generation

Casting the STLP as a partitioning problem as in Section 2.4 suggests devising a heuristic that can generate promising routes that work well in tandem. We may employ the above-described PSA to generate such route sets. The heuristic generates several routes, including those in the multiple initial solutions, those in the final solutions, and intermediate routes that are generated and then combined during the procedure. We collect all such routes and solve the set partitioning problem on this set. The solution returned by this set partitioning problem is necessarily of greater or equal quality than any of the final solutions produced by the PSA. We call this matheuristic approach PSA+Opt.

2.6 Computational Study

We now share results from a computational study of our proposed solution methodologies. We begin in Section 2.6.1 by introducing the test bed of problem instances we generated. We then compare the exact approaches and heuristics in Section 2.6.2. Finally, we present a sensitivity analysis of various parameters in Section 2.6.3.

2.6.1 Test Bed

To evaluate our methodologies, we generate a set of artificial instances designed to capture the environment in which they would be employed. All generated instances, as well as the source code for their generation, are available at `https://github.com/ericoden/STLP-data`.

We generate an instance with $|C|$ customers in the following way. First, $|N|$ (x,y) coordinates corresponding to pickup and delivery locations are sampled from a Gaussian mixture model

(GMM) with five equally-weighted components. Means of the components are drawn from a 2,000 mi. $\times$ 2,000 mi. grid. Covariance matrices are created by multiplying randomly generated 2×2 matrices (with values uniformly drawn from -200 to 200) with their own transpositions. We use clustered, rather than completely random, coordinates to mimic realistic shipper location data. The choice of five clusters is arbitrary, but the positions are set to give rise to cross-country distances between nodes. For a given shipper, the pickup and delivery locations are sampled from the GMM such that the Euclidean distance between the two is not below 300 miles. Thus, all shipments are in the mid- to long-haul range, which, according to our industry contact, is the context in which the rate-per-mile costs are most appropriate. For each shipper, the load is drawn uniformly from 8 to 44 linear feet, which represent the lower and upper bounds of load sizes described as most appropriate for the STL model (i.e., loads occupying 15-83% of a standard 53' trailer). Trailer capacity is set to 53 linear feet. The maximum delay factor is set to $\lambda = 2$, and the maximum number of customers a route can serve is set to $\tau = 4$. Demand levels for nodes are set so that demand levels are highest in the centers of clusters. This is done by calculating, for each node, the average distance to its five nearest neighbors. The nodes in the lowest third percentile are high-demand, the nodes in the top third percentile are low-demand, and the rest are medium-demand. A visualization of the location and demand data is presented in Figure 2.3. The per-mile fares, given the origin and destination demand levels, are set as presented in Table 2.2. The range of \$1 to \$4 for per-mile fares comes from our industry partner. We have also been informed that stop costs can range from \$50 to \$300 per stop; we use $\$175$ as our nominal value.

		Demand at End Node		
		Low	Medium	High
	Low	\$4	\$3.25	\$2.5
Demand at Start Node	Medium	\$3.25	\$2.5	\$1.75
	High	\$2.5	\$1.75	\$1

Table 2.2: Fare table for test bed.

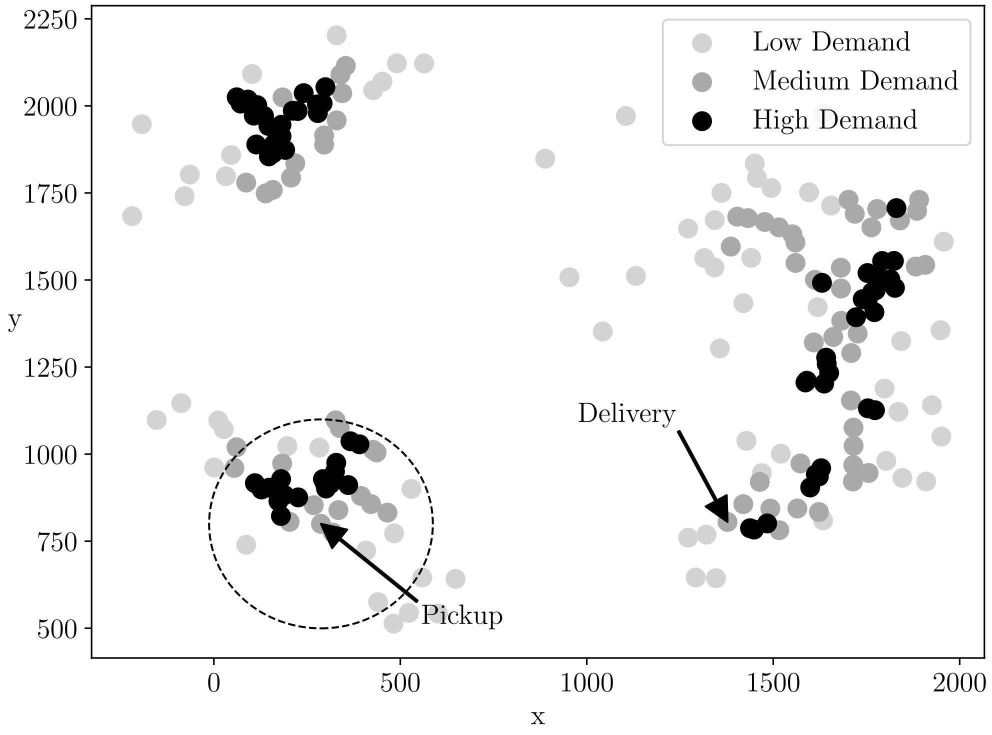


Figure 2.3: Example instance of a 100-customer STLP.

2.6.2 Results

Experiments were run on a computer with an Intel i5-7400 processor running at 3.00 GHz using 8 GB of RAM. All algorithms were implemented using Python 3.7.8. All linear programs were solved with Gurobi 9.5.1. In the following, we define the "gap" associated with a solution by the formula: $100 \cdot (z - z_{\text{best}})/z_{\text{best}}$, where z is the objective value of the solution, and z_{best}

is that of the best known solution. We define the "MIP gap" of a solution identified by a method that provides a lower bound on the optimal solution, LB, by the formula: $100 \cdot (z - LB)/LB$. All run times are in units of seconds.

We first evaluated the performance of the exact approaches: the arc-based MIP formulation (arc) presented in Section 2.3 and the branch-and-price algorithm (B&P) presented in Section 2.4. Both algorithms were run on a set of instances ranging from 5 to 10 customers, with 10 instances per problem size, and given a 60-second time limit. Average run times, as well as the MIP gap and gap for the arc-based formulation, are presented in Figure 2.3. B&P was able to find and prove the optimal solution within a second on all instances, while the arc-based formulation's run times quickly rose, reaching the time limit several times on the larger instances (it reached the time limit on six of the ten 10-customer instances). The MIP gap for the arc-based formulation also grew. However, in all except a single 10-customer instance, the upper bound was optimal.

	Arc Run Time		B&P Run Time		Arc MIP Gap		Arc Gap	
	Mean	Max	Mean	Max	Mean	Max	Mean	Max
n								
5	0.16	0.41	0.02	0.03	0	0	0	0
6	1.32	10.21	0.05	0.06	0	0	0	0
7	4.49	25.68	0.12	0.28	0	0	0	0
8	11.11	60.39	0.20	0.72	5.33	53.32	0	0
9	19.23	60.56	0.30	1.11	13.83	79.02	0	0
10	37.51	60.89	0.43	0.91	26.54	71.78	0.16	1.55

Table 2.3: Performance of the arc-based MIP formulation vs. B&P.

We then tested the B&P procedure on a set of larger instances ranging from 10 to 24 customers, with 100 instances per problem size. Table 2.4 presents the average and maximum run times for each fixed customer size. We observe that B&P, on average, takes only a minute on the largest (24-customer) instances in the set, but that it is possible for the run time on a given problem to near half an hour, due to repeated branching. We also assessed the performance of the heuristic of column generation restricted to the root node of the branch-and-price tree. We refer to this heuristic as CG. We include the average and maximum run times, Mip Gaps, and gaps for CG in Table 2.4. We observe CG takes under 30 seconds for all instances and on average under 7 seconds for the largest instances. Furthermore, in all but a single 21-customer instance, the upper bound at the root node was optimal. This demonstrates the bulk of the run time in B&P is spent proving optimality. Fitting a power function to B&P mean run times yields the estimate $1.64 \cdot 10^{-6} \cdot n^{5.34}$, and for CG, the estimate is $2.08 \cdot 10^{-5} \cdot n^{3.99}$. In addition to restricting column generation to the root node, we also attempted modifying the pricing problem so as to ignore the routes with more expensive fares. In particular, we deleted arcs of the form (s, i, f) from the pricing network, where f was among the more expensive fares (i.e., \$4 or \$3.25). The rationale for this restriction was that routes with high fares would not appear frequently in good solutions, and, thus, there is a low return on investment for exploring such routes. We denote this restricted form of root node column generation RCG, and we include the performances in Table 2.4. When we removed routes with high fares from consideration, run times modestly decreased. However, on average, solutions were 2% worse than the optimal solutions and could be as far as 30% from optimal. Given the performance of this heuristic compared to the PSA described in Section 2.5.2, we did not evaluate it further.

We then evaluated the performance of the heuristics described in Section 2.5: the baseline (Baseline), the parallel savings algorithm (PSA), as well as the matheuristic of solving the set partitioning problem, restricted to the routes generated by the PSA (PSA+Opt). We used the same instances as those in Table 2.4 and recorded the gap and run time for each method. We have access to the optimal solutions for such instances, and hence the optimality gaps for our

	CG MIP Gap		CG Gap		RCG Gap		B&P Run Time		CG Run Time		RCG Run Time	
	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
n												
10	0.36	9.39	0	0	2.71	30.88	0.31	2.61	0.24	0.34	0.15	0.22
11	0.25	4.21	0	0	2.85	21.21	0.40	1.32	0.35	0.61	0.21	0.31
12	0.24	6.25	0	0	2.69	25.41	0.61	5.02	0.46	0.67	0.28	0.50
13	0.26	4.55	0	0	1.72	29.80	0.84	4.39	0.62	0.86	0.39	0.49
14	0.48	4.90	0	0	2.74	20.08	2.15	72.39	0.81	2.08	0.57	1.72
15	0.36	2.74	0	0	2.86	18.84	2.03	25.35	1.02	2.08	0.70	1.28
16	0.57	7.27	0	0	2.44	15.82	3.95	72.48	1.32	2.07	0.89	1.57
17	0.36	5.22	0	0	2.79	25.49	3.65	78.75	1.76	3.05	1.06	2.08
18	0.38	3.65	0	0	2.02	12.19	5.07	125.24	2.20	3.01	1.38	2.29
19	0.41	3.38	0	0	2.82	14.90	9.99	382.61	2.68	4.55	1.65	2.27
20	0.35	3.35	0	0	2.87	16.39	7.49	72.52	3.09	5.28	2.03	4.07
21	0.44	4.04	3.4e-06	3.4e-04	3.00	31.28	20.79	1,093.10	3.80	8.86	2.47	5.55
22	0.53	5.14	0	0	2.50	17.87	31.40	1,362.13	4.58	10.21	3.03	9.74
23	0.49	4.76	0	0	2.14	11.09	45.54	1,770.96	5.68	15.58	3.87	12.52
24	0.40	3.13	0	0	2.57	17.44	26.15	635.86	6.78	25.61	4.74	21.69

Table 2.4: Run times of column generation methods.

heuristics, due to the B&P algorithm. The results, averaged over 100 instances for each problem size from $n = 10$ to $n = 24$, are presented in Table 2.5. We observe the gap for the baseline heuristic grows from 12% to 16% on average and can produce solutions as high as 50% from optimal. The gap for PSA also grows, but only from 0.05% to 1.20%. Furthermore, the gap is within 9% for all instances. PSA+Opt performs better still, with the average gap rising from 0% to around 0.2% and never above 4%. For the largest instances, both PSA and PSA+Opt complete within 3 seconds.

We then investigated the performance of the baseline, CG and PSA+Opt heuristics on larger instances of the problem. Table 2.6 presents the performances of the two approaches for problem sizes ranging from 30 to 90 customers, in steps of 10, with five instances per problem size. The best solutions are found by CG in all instances. Furthermore, the lower bounds produced by CG demonstrate that the solutions found are, on average, within 1% of optimal. The average run times for CG exceed an hour for the 90-customer instances and took 8 hours in one instance.

	Baseline Gap		PSA Gap		PSA+Opt Gap		PSA Run Time		PSA+Opt Run Time	
	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
n										
10	12.34	43.82	0.05	3.10	0	0	0.16	0.40	0.18	0.45
11	10.78	38.23	0.10	3.01	0.01	1.17	0.21	0.49	0.23	0.53
12	11.64	47.68	0.10	7.20	0	0	0.27	0.57	0.29	0.61
13	12.99	39.66	0.18	2.80	0.03	1.62	0.36	0.75	0.38	0.77
14	13.54	41.93	0.11	3.06	0	0	0.45	0.89	0.48	0.92
15	13.63	50.10	0.19	4.49	0.04	3.19	0.53	0.93	0.59	1.03
16	13.29	42.01	0.42	5.23	0.02	1.50	0.66	1.17	0.70	1.17
17	13.71	50.95	0.56	6.02	0.01	0.57	0.68	1.33	0.72	1.35
18	12.83	40.11	0.53	8.85	0.11	1.76	0.83	1.45	0.86	1.50
19	15.97	36.68	0.75	4.76	0.09	3.09	1.02	2.04	1.06	2.07
20	14.62	47.22	0.72	6.13	0.13	2.67	1.19	2.00	1.19	1.92
21	15.92	40.83	0.85	6.06	0.11	3.67	1.32	2.28	1.36	2.60
22	14.58	29.96	0.70	4.81	0.14	2.15	1.46	2.29	1.51	2.26
23	14.84	38.96	0.92	4.81	0.20	3.22	1.60	2.45	1.65	2.53
24	16.76	43.05	1.20	8.48	0.18	1.97	1.74	2.63	1.82	2.75

Table 2.5: Performance of heuristic approaches.

The solutions produced by baseline algorithm have, on average, about 23% larger objective value. Meanwhile, those found by PSA+Opt are, on average, within 3%, and always within 5%. Furthermore, the solutions found by PSA+Opt were always found within a minute. Overall, we conclude the PSA+Opt heuristic is an effective and scalable solution methodology for the STLP.

	Baseline Gap		CG MIP Gap		CG Gap	PSA+Opt Gap		CG Run Time		PSA+Opt Run Time	
	mean	max	mean	max	max	mean	max	mean	max	mean	max
n											
30	19.14	37.02	0.32	1.22	0	0.11	0.34	15.83	21.35	3.30	5.20
40	23.72	30.12	0.69	1.20	0	1.25	4.22	50.82	56.56	7.85	8.90
50	22.92	31.73	0.35	0.56	0	1.75	2.92	215.11	494.16	12.53	14.58
60	23.64	30.80	0.58	1.07	0	1.06	1.86	1,925.31	5,541.06	17.09	22.15
70	25.21	30.68	0.11	0.33	0	2.28	3.64	2,013.65	3,795.46	26.47	32.25
80	24.75	29.29	0.31	0.71	0	1.27	2.73	2,790.91	4,844.06	36.34	44.77
90	23.30	32.42	0.37	1.03	0	2.66	4.88	8,776.56	28,958.66	44.79	52.02

Table 2.6: Performance of heuristic approaches on larger instances.

2.6.3 Sensitivity Analysis

To learn more about the STL model, we performed a sensitivity analysis on various problem parameters. Figure 2.4 presents the results of an experiment evaluating the dependence on the maximum delay factor, λ. The PSA+Opt heuristic was used to solve fifty 100-customer instances, with λ varying from 1 to 4 in steps of 0.25. We plot the average objective value, as well as the average number of customers served by 1-, 2-, 3-, and 4-routes. Naturally, when $\lambda = 1$, no pooling is possible, and all routes are 1-routes. When λ is increased to 1.25, we observe a jump in the number of 2-routes as well as a significant (29%) drop in the objective value. However, we quickly observe diminishing returns for further increases of λ; increasing from 1.25 to 1.5 only decreases the objective by 6%. As λ rises further, the additional flexibility allows for more 3- and 4-routes. However, even for $\lambda = 4$, a customer is most likely to be pooled with a single other customer. This is likely a consequence of the relatively large load sizes. The average load occupies around half of a trailer, and, thus, pools of three or four customers are necessarily rare.

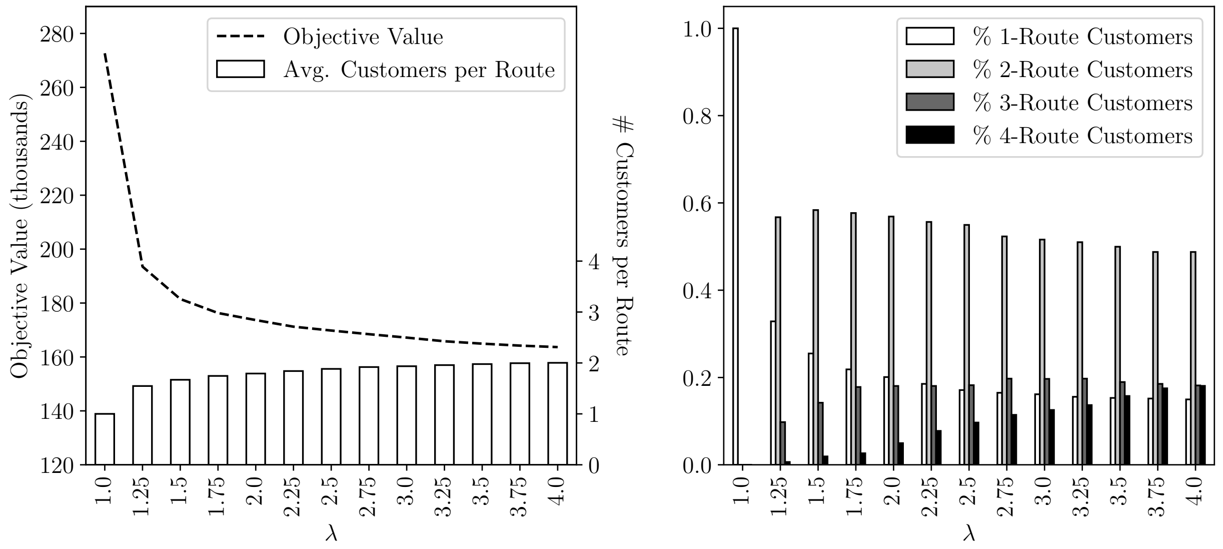


Figure 2.4: Sensitivity analysis of the delay factor, λ.

Figure 2.5 presents the results of a similar experiment where the stop cost, E, was varied from 100 to 1,000 in steps of 100. We observe that the relative proportions of the 1-, 2-, 3-, and 4-route customers evolve slowly as the stop costs are increased. Rather than encouraging alternative poolings and sequences, the increase in stop costs merely increases the cost of the (near) optimal routes, resulting in an almost perfectly linear trend. However, we do observe that the number of 1-routes gradually increases while the number of 2-, 3-, and 4-routes gradually decreases.

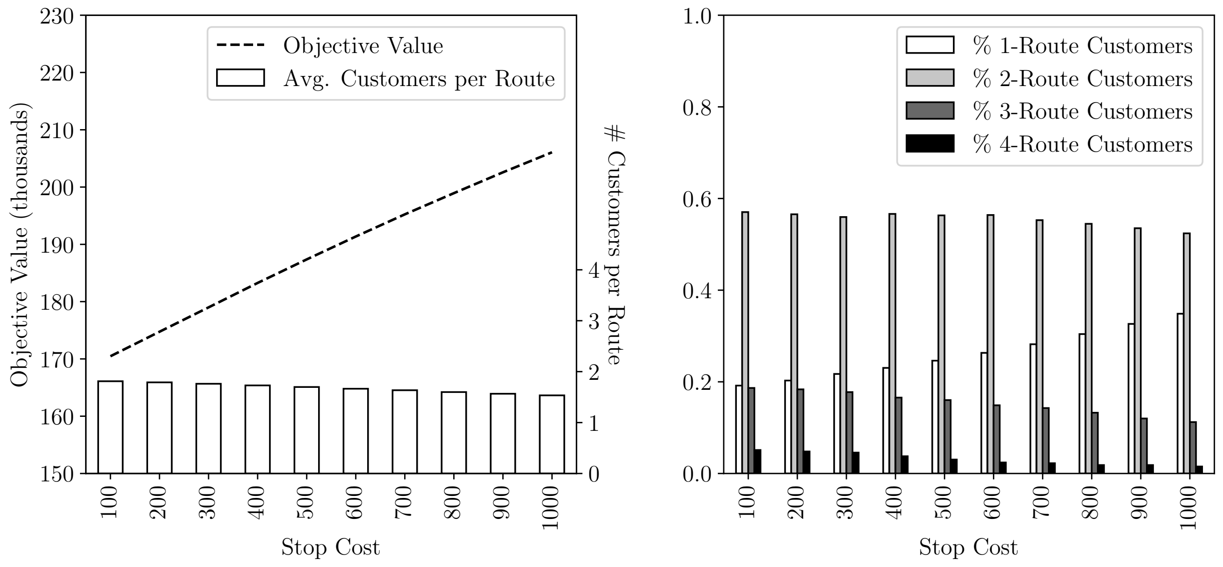


Figure 2.5: Sensitivity analysis of the stop cost, E.

Setting a maximum number of customers per route, τ, significantly reduces the number of feasible routes, ignoring capacity and temporal constraints. However, experiments suggest raising τ from the nominal value of 4 will not lead to lower costs overall. Figure 2.6 presents the average objective values over 50 instances for problem sizes from 70 to 100, with τ set to $1, 2, 3,$ and 4. We observe the rapidly diminishing returns from increasing τ. This is largely due to the combined effect of the routing constraints and the relatively large size of the freight, as routes serving several customers are relatively rare under such settings.

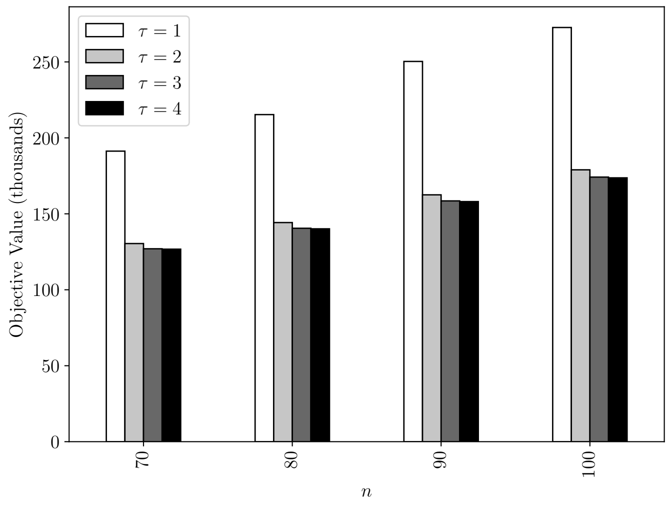


Figure 2.6: Sensitivity analysis of the maximum number of customers per route, τ.

We demonstrate the dependence on the customer load sizes with the experiment presented in Figure 2.7. For each value of $\alpha \in \{0.1, 0.2, \dots, 0.9, 1\}$, and for fifty 100-customer instances each, the load for each customer was multiplied by α and the problem solved. A customer's load is therefore between 8α and 44α linear feet. The results are averaged over each value of α. We observe that for lower values of α lead to larger poolings. For $\alpha \leq 0.6$, 4-routes become the dominant variety. Interestingly, 3-routes are never dominant.

Next, we investigated the effect of the LIFO constraints on the problem. Specifically, we wanted to determine the "price" of the constraints. We identified solutions by relaxing the constraint in PSA+Opt. Specifically, when computing the savings associated with combining two routes, we also considered combinations that do not satisfy LIFO constraints. In doing so, the number of possible combinations significantly increases. For example, ignoring capacity and time constraints, a route serving four customers has 120 feasible sequences with LIFO constraints

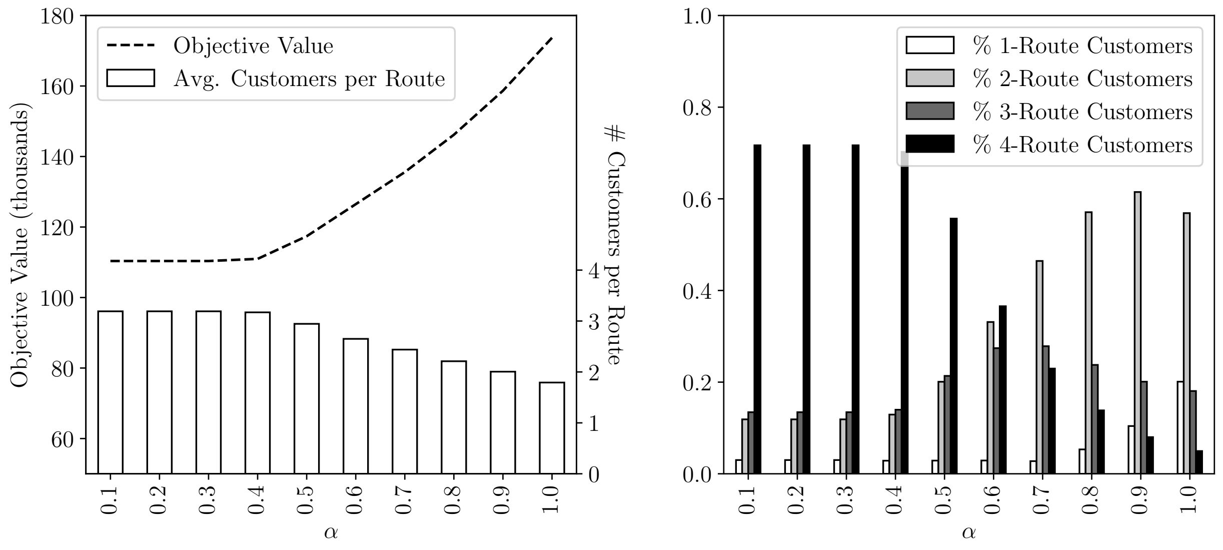


Figure 2.7: Sensitivity analysis of the load sizes.

and 1,776 without. This significantly increases the run time of the `GetSavings` procedure in PSA+Opt, and, in order for the algorithm to scale, it is advisable to approximate the maximum savings rather than compute it exactly. We developed such an approximation scheme for the PSA, which we present in Appendix A.2. We also investigated the effect of allowing intermediate deadheads (i.e., allowing trailers to be empty during the route). This similarly expands the space of solutions, and we once again used the PSA+Opt procedure with a modified `GetSavings` function to identify solutions, also described in Appendix A.2.

We then ran PSA+Opt on the problem with all constraints, with all except LIFO constraints, and with all except deadhead constraints (the latter two using the approximation scheme) on instances ranging from 30 to 100 customers, with 50 instances per problem size. Figure 2.8 presents the average objective values, as well as the average number of customers per route in both settings. No LIFO refers the problem with LIFO constraints relaxed, and DH refers to the problem with the prohibition on deadheading relaxed. We observe that the price of the deadheading constraints appears to be greater than that of the LIFO constraints. The cost of LIFO constraints rises with increasing n, from 21% when $n = 30$ up to 27% when $n = 100$. Meanwhile, the cost of the deadheading constraints is consistently around 33% for all n. We observe that the average number of customers per route in all three settings is relatively unresponsive to n, remaining near 1.75, 2.3, and 3.2 for the original problem, the problem with LIFO constraints relaxed, and the problem with deadhead constraints relaxed, respectively.

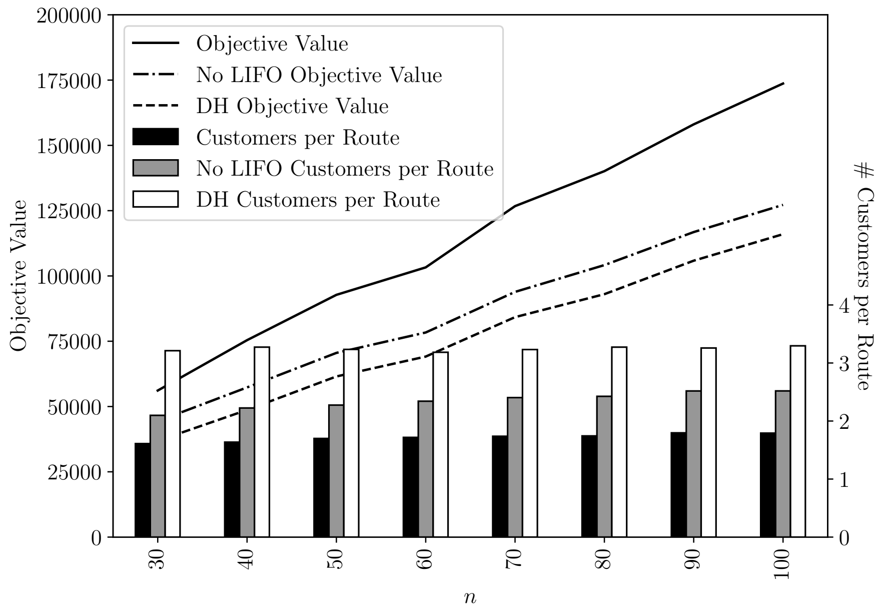


Figure 2.8: Sensitivity analysis of the constraints.

As noted in Section 2.1, relaxation of LIFO constraints is likely unrealistic, given the one-touch policy of the STL provider, the relatively large size of the shipments most appropriate for STL transport, and the relative rarity of trucks capable of side-loading. However, the deadhead constraints arise from a reluctance for carriers to accept such routes, due to their perceived inefficiency. Based on the results of the experiment presented in Figure 2.8, we recommend that deadheading routes be seriously considered by STL carriers.

We summarize our managerial insights as follows:

- Increases of λ have rapidly diminishing returns. Simply raising λ from 1 to 1.25 leads to 29% drop in objective value, but increasing from 1.25 to 1.5 only leads to a 6% drop.
- The better solutions generally remain so, even after increasing the stop cost.
- 2-routes are the dominant variety, even with larger values of λ. This can be explained by the relatively large size of the freight. Furthermore, there is little benefit from considering routes serving more than 4 customers.
- The possible savings from allowing deadheading within the routes are dramatic, and exceed those from relaxing LIFO constraints. As the prohibition of deadheading strikes us as somewhat artificial, efforts to normalize such routes in the STL industry could make sense.

2.7 Conclusions

We have presented a shared truckload design problem relevant to an emerging freight paradigm, which offers the savings associated with freight pooling as well as the reliability and minimal risk associated with direct shipping. We offer a mathematical formulation of the problem as well as a corresponding branch-and-price algorithm. We demonstrate that, given parameter settings advised by an industry contact, the algorithm is able to produce and prove optimal solutions within an hour for instances with 24 customers, with typical run times of less than a minute. Furthermore, we demonstrate that the heuristic of restricting column generation to the root node can produce solutions provably within 1% of optimality within an hour for instances with up to 80 customers. We also present a parallel savings algorithm, as well as an optimization-based enhancement. We demonstrate the resulting heuristic can produce solutions within 3%

of optimality within a minute for 90-customer instances. Using this heuristic, we conduct a sensitivity analysis of the problem and illustrate the diminishing returns associated with relaxing the quality of service constraints, the relative insensivity to stop costs, the dominance of routes serving only two customers, and the savings possible due to relaxation of LIFO and deadheading constraints.

There are a number of further directions to pursue in this domain. While efforts were made to simulate realistic instances, it would be useful to evaluate our approaches on real freight data. In particular, it would be useful to compare the costs and benefits of the LTL and STL paradigms from the perspective of the coordinating entity as well as that of the shippers and carriers. In addition, there may be important practical considerations unique to STL that are not included in our model. For example, according to our industry contact, there is an execution risk associated with pooling. Specifically, the greater the time before the last pickup, the greater the risk of a trucker experiencing service issues that prevent them from picking up remaining shipments. The pickups that remain must then be rescued (i.e., serviced alone). Extending our model to incorporate such concerns is a possible next step. Finally, there are further areas for improvement of the PSA+Opt algorithm. In particular, the initial solutions were generated by a fairly simple procedure (i.e., producing solutions within a small neighborhood of the trivial solution). We experimented with other initialization schemes, and found little improvement upon the one we employed. However, it may be worth continuing to investigate initialization schemes that could lead to the generation of a more diverse pool of high-quality routes. We believe the synthesis of the PSA and the set-partitioning formulation is a promising solution methodology which may also serve other applications.

Chapter 3: The Rendezvous Vehicle Routing Problem

3.1 Introduction

The e-commerce industry has steadily grown over the past decade, with the share of total sales in the U.S. rising from 4.5% in 2011 to 14.3% in 2020 [according to 43]. The rise of online sales has, in turn, popularized same-day delivery, nullifying the instant-gratification advantage of brick-and-mortar retail options. Indeed, same-hour shipping has entered the lexicon [44]. This frontier of convenience presents a number of formidable logistical challenges which have received considerable attention in the operations research community [see the survey by 45].

In this chapter, we consider a problem posed to us by a leading provider of logistics software in the package delivery industry. By some accounts, last-mile delivery accounts for 41% of total supply chain costs [see 46]. With the increased volume of same-day delivery, and the proportion of costs that last-mile delivery represents, it is critical for delivery companies to provide for same-day delivery efficiently. Typically, parcel delivery companies dedicate a distinct fleet of vehicles (that we will refer to as shuttles) for same-day delivery. Because same-day delivery takes place in the afternoon and evening (after a sufficient number of orders have accumulated in the morning) the shuttles leave on their routes later in the day than the rest of the vehicles involved in regular parcel delivery (that we refer to as trucks). The question posed to us, and the opportunity it represents, relates to allowing for some type of synchronization between the trucks and shuttles [specifically, operational synchronization, as described by 47]. The idea is to allow shuttles to hand off or transfer packages to trucks for customers on a truck's route that have not yet been visited. This would free the shuttle from visiting those customers, thereby reducing costs and allowing for larger volumes of same-day delivery with a fixed number of shuttles.

In this work, we seek to design efficient shuttle routes that aim to minimize the distance traversed by the shuttles. With the proposed algorithm, we can then quantify the savings this routing strategy could generate when compared to the current status quo (i.e., having shuttle(s) visit all of the same-day delivery location). We begin by explicitly defining our problem and exploring the relevant literature in Section 3.2. We then develop a number of approaches to address the problem in Section 3.3, including an arc-based formulation (3.3.1), a path-based formulation (and corresponding column generation scheme) (3.3.2), and a heuristic (3.3.3). Next, we evaluate the relative strengths of the three approaches, as well as the effectiveness of truck-shuttle synchronization in different scenarios, in Section 3.4. We conclude, in Section 3.5, that our column generation approach is an efficient exact algorithm for designing shuttle routes for instances with up to 200 nodes, and that our heuristic provides excellent solution quality for larger instances, with as many as 1,000 nodes. For the test instances considered in this chapter, we demonstrate that shuttle-truck synchronization results in as much as a 77% savings in transportation costs, and, thus, should be seriously considered by logistics providers in the same-day delivery industry.

3.2 Problem Statement

We assume truck routes partition the entire customer network. That is, each possible customer is assigned to a unique truck, which weaves through its daily route regardless of the particular day's deliveries. This is motivated by the example of the US Postal Service (USPS), where every potential customer is located on a mail delivery route. The situation is similar in the package delivery industry (e.g., UPS, Fedex, and Amazon) where routes remain roughly the same on a daily basis even if there is no package delivery at a specific customer's location on a given day [see 48]. For the USPS, UPS, Fedex, Amazon, and other major logistics providers, the density of customer locations within a neighborhood is sufficiently high to necessitate that nearly every street, if not every home, is visited on a daily basis. The impact of this observation is that nearly every last-minute service location (LMSL) will lie on some truck's route, and, thus, the trucks will not have to alter their routes in order to serve them. In the rare cases where an LMSL cannot be easily inserted into a truck route or where the LMSL has already been passed by its truck at the beginning of the problem, we require the shuttle to make the delivery. To be clear, this is not an artificial problem. Rather, it was posed recently by one of the major logistics providers in the U.S.

Given a fleet of trucks, a depot, fixed truck routes (each defined as a sequence of locations, starting and ending with the depot), a fleet of shuttles, a set of LMSLs that lie somewhere on the truck routes, and current locations of each truck, we seek the minimum distance shuttle routes, each starting and ending with the depot, such that each LMSL is serviced. An LMSL can be serviced either by meeting with a truck that will visit the LMSL later on, or by visiting the LMSL itself. Such meetings can only happen at service locations (where the truck will be stopped anyway), rather than at any point on the truck's route. We do not consider capacities (as same-day delivery packages tend to be small and truck capacity is generally not the limiting factor), nor service times. A shuttle must wait for a truck to arrive, but a truck cannot wait for a shuttle. A delivery is *direct* if a shuttle visits the LMSL itself in order to deliver the package; a delivery is *indirect* if the package is delivered by a truck, which has met with a shuttle at a previous stop. Figure 3.1 visualizes this distinction. LMSL A and B are both direct deliveries, since the shuttle visits both of them. Meanwhile, LMSL C is an indirect delivery, since LMSL C is visited by a truck that has previously met with the shuttle (at the rendezvous point). All other nodes, including the rendezvous point, are regular service locations. Altogether, we call this problem the Rendezvous Vehicle Routing Problem (RVRP). A sample solution to the RVRP with four trucks and one shuttle is presented in Figure 3.2. The gray paths correspond to truck routes, with arrows indicating their direction. The black node is the depot, and the red nodes are LMSLs. The orange path is the optimal shuttle route, which serves all LMSLs either directly or indirectly, with the minimum distance traveled. It first makes a nearby direct delivery, then makes a rendezvous with the truck on the top right to hand over four packages. It then meets with the truck on the top left to hand over one package, then makes three direct deliveries, then meets with the truck on the lower right to hand over one package, and finally returns to the depot. Note that each time a rendezvous occurs, the truck and shuttle appear to have traveled roughly the same distance along their respective routes.

We note an important subtlety in the choice of objective, specifically, to minimize the length of the shuttle route(s). Though this is a natural choice when the problem is viewed as a variant of the traditional vehicle routing problem, it may be more important, in practice, to minimize the *time* before the shuttles return to the depot. Optimal solutions for each objective function may be

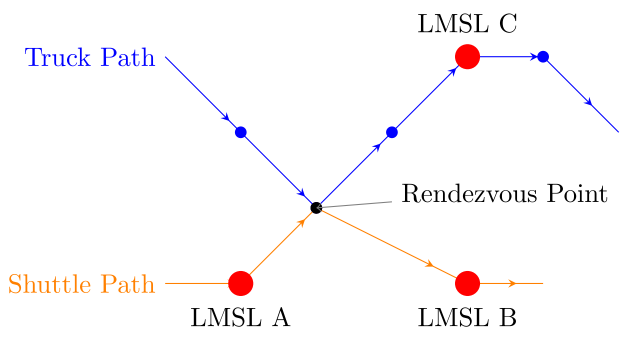


Figure 3.1: Comparison of direct and indirect deliveries.

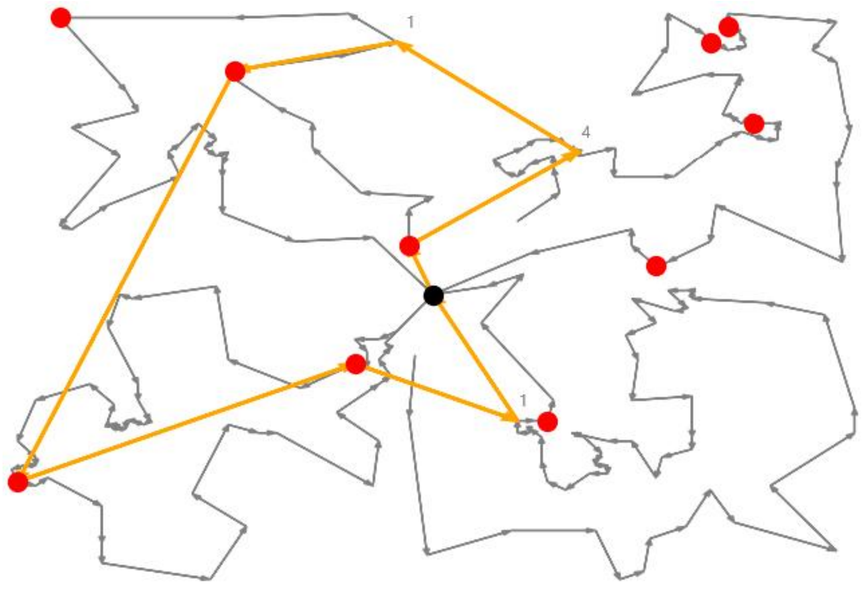


Figure 3.2: Sample solution to the RVRP with four trucks and one shuttle.

different, due to the possibility of a shuttle waiting at a node in anticipation of a truck. Yet another objective is the minimization of the number of shuttles necessary to achieve a certain quality of service (e.g., each LMSL delivered prior to some deadline). Some of the structures we develop in this chapter can be adapted for each of these cases in a straightforward manner. However, motivated by reducing the carbon footprint of logistics providers, in our work, we concentrate on the distance-minimization version of the problem.

To our knowledge, our work is the first to study this delivery strategy. For that reason, we focus on the simplest version of the problem. That is, we study the static and deterministic version (i.e., all information is known at the outset, and all quantities are nonrandom). However, in practice, building stochastic travel times into the model seems advantageous. In particular, it is easy to imagine a long shuttle path made necessary by a single missed connection. Such considerations, however, can be incorporated into the deterministic model by introducing, for each rendezvous point, a deadline *prior* to the truck's arrival by which a shuttle must arrive for a rendezvous to occur. The length of the interval between the deadline and the truck arrival can be the same for all rendezvous points, or can vary (perhaps larger for stops further along a truck's path). Furthermore, the length of the interval can be tuned by a practitioner to balance robustness to travel time fluctuations with solution quality.

We observe that the RVRP is NP-Hard. Consider an instance of the RVRP with a single shuttle, in which each truck has already passed all of the LMSLs on its route. Then, the distance-minimizing shuttle tour that serves each LMSL is simply the solution to the traveling salesman problem on the complete graph defined by the LMSLs and the depot.

3.2.1 Literature Review

The RVRP can be viewed as a VRP with multiple synchronization constraints (VRPMS), a broad class of problems described and categorized in the survey presented by Drexl [47]. The survey defines a VRPMS as a "vehicle routing problem in which more than one vehicle may or must be used to fulfill a task". The paper establishes a taxonomy of such problems, describing five kinds of synchronization constraints: task, operation, movement, load, and resource. The first two apply to the RVRP. Task synchronization refers to the specification that each task must be performed exactly once by one or more suitable vehicles. In our case, each LMSL must be served by either a truck or a shuttle. Operation synchronization refers to some requirement that, in order to perform some operation between them, multiple vehicles must coordinate in time and/or space. This includes the requirement of meeting at a node for truck-shuttle transshipment in our problem. An archetypal case (that is, one exhibiting all five varieties of synchronization constraints) is presented in Drexl [49]. Fink et al. [50] solve another archetypal case exactly with a branch-and-price scheme. In the RVRP, synchronization arises only from the possibility of transshipment of deliveries between trucks and shuttles.

The inclusion of transshipment capabilities in vehicle routing contexts has received attention in recent years, with several papers exploring an extension of the pickup and delivery problem where there exists a set of transshipment locations at which vehicles can transfer their loads. Cortés et al. [51] present an MIP formulation of the problem, as well as an exact approach based on Benders' decomposition. Rais et al. [52] present a formulation where the flows of vehicles and loads are distinct and are linked using multicommodity flows. Heuristic approaches include that of Mitrović-Minić and Laporte [53], who present a two-phase heuristic, with a construction phase and an improvement phase. Masson et al. [54] and Sampaio et al. [55] utilize an adaptive large neighborhood search algorithm, the latter to solve in the context of crowdshipping. Thangiah et al. [56] heuristically solve the problem, where transshipments are allowed at any pickup or delivery location. We note, however, the problems described in these six papers do not apply in the same-day delivery context.

The RVRP is designed to produce solutions in the context of same-day delivery. Ulmer et al. [57] consider accommodating same-day orders by modifying truck routes to include preemptive depot returns (i.e., returning to the depot prior to delivering all cargo). They use an approximate dynamic programming approach to develop routes that maximize the number of served requests, where values of decisions are generated using approximate value iteration. Their method enables nearly instantaneous decision making at run time. Voccia et al. [58] seek to maximize the number of same-day deliveries, where customers also have time windows. They use a scenario-sampling approach, where the action taken at each epoch is determined by the action that is representative of the solutions in various scenarios. They demonstrate situations in which strategic waits at the depot in anticipation of demand can increase the total number of customers served. Azi et al. [59] solve a similar problem, except where the objective is to maximize customer-specific profits. They too use a scenario-sampling approach, but limit the route length by a fixed parameter. Klapp et al. [60] also solve a similar problem, except that all requests are on a line. In their work, an approximate linear programming approach is employed. In none of the above papers is transshipment considered as a scheme in the same-day delivery setting, which is the principle contribution of our work.

3.3 Solution Approaches

We now present three approaches developed to address the RVRP. We begin with an arc-based formulation in Section 3.3.1, then a path-based formulation (with an associated column generation algorithm) in Section 3.3.2, and finally a heuristic in Section 3.3.3.

3.3.1 Arc-Based Formulation

Let $N = \{0, 1, \ldots, n, n+1\}$ denote the set of nodes in the problem (including the depot, all LMSLs, and all regular service locations). Let node 0 and node $n+1$ denote the depot at the beginning and end of the problem, respectively (these may be at the same location). Let $L \subset N$ be the set of LMSLs. Let $R \subseteq N$ be the set of nodes that lie on a truck route. For each $r \in R$, let $T_r \in \mathbb{R}^+$ be the time the truck arrives at that node (this is a known quantity). Let $d_{ij} \in \mathbb{R}^+$ be the distance from node $i \in N$ to node $j \in N$, such that $d_{ii} = d_{0,n+1} = 0$ for all $i \in N$, and $d_{ij} > 0$ otherwise, and let the matrix $\mathbf{d}$ satisfy the triangle inequality. Let $x_{ij} \in \{0,1\}$ be the decision variable which denotes whether a shuttle travels from node $i \in N$ to node $j \in N$. Let the decision variable $p_r \in \{0,1\}$ for $r \in R$ denote the use of node r as a rendezvous point. For each r in R, let $S_r \subseteq L$ be the set of LMSLs that can served by making a rendezvous with a truck at node r, but not including node r if node $r \in L$. Thus, nodes in S_r are all serviced indirectly by the visit to r. Let the decision variables $t_i \in \mathbb{R}^+$ and $u_i \in \mathbb{R}^+$ represent the time a shuttle arrives at node i and departs from node i, respectively. If node i is not visited by a shuttle, t_i and u_i are meaningless. Let $V \in \mathbb{Z}^+$ be the number of shuttles available. The following mixed-integer linear program models the RVRP:

$$\min \sum_{i \in N} \sum_{j \in N} d_{ij} x_{ij} \tag{3.1}$$

$$\text{s.t.} \sum_{j \in N} x_{jl} + \sum_{r \in R | l \in S_r} p_r = 1 \qquad \forall l \in L \tag{3.2}$$

$$\sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = \begin{cases} V & i = 0 \\ -V & i = n+1 \\ 0 & i \in \{1, \dots, n\} \end{cases} \qquad \forall i \in N \tag{3.3}$$

$$p_r \leq \sum_{j \in N} x_{jr} \qquad \forall r \in R \tag{3.4}$$

$$u_i + d_{ij} - (M + d_{ij})(1 - x_{ij}) \leq t_j \qquad \forall (i,j) \in N \times N \tag{3.5}$$

$$t_r - M(1-p_r) \leq T_r \qquad \forall r \in R \tag{3.6}$$

$$T_r p_r \leq u_r \qquad \forall r \in R \tag{3.7}$$

$$t_i \leq u_i \qquad \forall i \in N \tag{3.8}$$

$$\mathbf{t}, \mathbf{u} \geq 0, \mathbf{x}, \mathbf{p} \in \{0,1\} \tag{3.9}$$

The objective 3.1 minimizes total travel distance for the shuttles. Constraints 3.2 ensure each LMSL is either visited directly (first term) or is visited indirectly by meeting with a truck that will visit with the LMSL (second term). Constraints 3.3 ensure the conservation of flow through the network. Constraints 3.4 ensure that node r is used as a rendezvous point only if a shuttle visits it. Constraints 3.5 capture the propagation of shuttle arrival times (where M is a sufficiently large number). Constraints 3.6 require shuttle arrival prior to truck arrival at node $r \in R$ if r is to be used as a rendezvous point, and constraints 3.7 further impose that the shuttle must wait for the truck. Constraints 3.8 impose that departures follow arrivals. Constraints refrvrp:eq:domains establish the domains of the decision variables. The following proposition provides a value for M.

Proposition 1. *Let $TST \in \mathbb{R}^+$ be the length of the minimum-distance tour which starts at node 0, ends at node $n+1$, and visits each node in L. If $M = TST + \max_{r \in R} T_r$, then constraints 3.5 and 3.6 are valid.*

Proof. Let $(\mathbf{x}, \mathbf{p}, \mathbf{t})$ be an optimal solution to the RVRP. Note $t_{n+1} \geq t_i$ for all $i \in N$. If the optimal solution does not rendezvous with a truck, then $t_{n+1} = TST$. If the optimal solution does rendezvous with a truck, then $t_{n+1} \leq T_r + TST$, where $r \in R$ is the final truck rendezvous node on the route. Thus, for all $i, j \in N$,

$$t_i - t_j \leq t_{n+1} \leq TST + \max_{r \in R} T_r = M$$

and for all $r \in R$,

$$t_r - T_r \leq t_{n+1} \leq TST + \max_{r \in R} T_r = M$$

□

We may derive a set of valid inequalities from the observation that, in an optimal solution, each truck is intercepted by a shuttle at most once.

Proposition 2. *Let* $(\mathbf{x}, \mathbf{p}, \mathbf{t})$ *be an optimal solution to the RVRP. Let* K *be the set of truck routes. Let* $R_k \subseteq R$ *be the set of rendezvous points on truck route* $k \in K$*. We have*

$$\sum_{r \in R_k} p_r \leq 1 \quad \forall k \in K \tag{3.10}$$

Proof. Suppose 3.10 does not hold. Then, there is some $k \in K$ and $i, j \in R_k$ such that $p_i, p_j = 1$. Either $S_i \subseteq S_j$ or $S_j \subseteq S_i$. Without loss of generality, suppose $S_j \subseteq S_i$. If $j \in L$, then $j \in S_i$. By 3.4, there is some $a \in N$ such that $x_{aj} = 1$, and by 3.3, there is some $b \in N$ such that $x_{jb} = 1$. Setting $x_{aj}, x_{jb}, p_j = 0$ and $x_{ab} = 1$ is feasible and, by the triangle inequality, decreases the objective value. Thus, 3.10 must hold. □

Therefore, we can include constraints 3.10 in the formulation.

3.3.1.1 Extensions

Our formulation can be modified to capture other elements that may be necessary to include in some settings. We shall explicitly describe two important extensions here. Firstly, we observe that the time required to serve an LMSL and the time required to transfer a package from a shuttle to a truck may be too large to ignore. Let $h_r \in \mathbb{R}^+$ be the time required to transfer the packages at rendezvous point $r \in R$. Let s_l be the time required to make the direct delivery at node $l \in L$, and let $s_i = 0$ for $i \in N \backslash L$. We must decide whether the truck or the shuttle delivers the package if the rendezvous node is also an LMSL. We shall assume the truck makes the delivery. The arc-based formulation is modified by replacing constraints 3.5, 3.7 and 3.8 with

$$\begin{aligned} t_i + s_i(1 - p_i) + h_i p_i &\leq u_i \qquad \forall i \in R && (3.11)\\ t_i + s_i &\leq u_i \qquad \forall i \in N \backslash R && (3.12)\\ (T_r + h_r)p_r &\leq u_r \qquad \forall r \in R. && (3.13) \end{aligned}$$

Constraints 3.11, and 3.12 ensure the difference between arrival time and departure time is at least long enough for the required service (delivery or transfer). Constraints 3.13 impose that a shuttle meeting a truck for a rendezvous at r cannot depart until h_r units of time after the truck's arrival.

Secondly, we may wish to respect a deadline for each truck's return to the depot, given the additional load from a shuttle. Since a given truck is only met at most once by a shuttle, a transfer of packages is only infeasible if the additional load from that single rendezvous causes it to return after the deadline.

Therefore, we may implicitly respect the deadlines in the following way. For each rendezvous point, r, we enumerate the $Q_r \in \mathbb{Z}^+$ subsets of $S_r \subseteq L$, $\{S_r^q\}_{1 \leq q \leq Q_r}$, such that the additional load on the truck does not exceed the truck's capacity, and such that $S_r^a \not\subseteq S_r^b \; \forall a, b \in \{1, \ldots, Q_r\}$. That is, we enumerate all sets of packages we could transfer at r such that no one set is a subset of another, and such that the corresponding increase in the truck's arrival time at the depot is within the deadline. These sets can be computed *a priori*. We note that we may relax our assumption that all potential LMSLs lie on truck routes, as we can include the time to deviate from the original route to accommodate an LMSL. In this setting, an LMSL l is in S_r if it is possible to transfer the package at r and for the truck to visit l, even if l is not on the original truck route.

The arc-based formulation is modified as follows. Let $p_r^q \in \{0, 1\}$ denote the use of rendezvous node $r \in R$ to transfer the LMSLs $S_r^q \subseteq L$, where $q \in \{1, \ldots, Q_r\}$. We replace the constraints:

$$\sum_{j \in N} x_{jl} + \sum_{r \in R : l \in S_r} p_r = 1 \qquad \forall l \in L$$

with

$$\sum_{j \in N} x_{jl} + \sum_{r \in R} \sum_{q \in Q_r : l \in S_r^q} p_r^q = 1 \qquad \forall l \in L$$

and include the constraints:

$$p_r^q \leq p_r \qquad \forall q \in Q_r, r \in R.$$

We can include both of the above considerations simultaneously. If the time to transfer packages at rendezvous point r is a function of the set of packages itself, small modifications to constraints 3.11 and 3.13 are necessary. Letting h_r^q be time required to transfer the set of packages S_r^q at rendezvous point r, we instead include the constraints:

$$\begin{aligned} t_i + s_i(1 - \sum_{q \in Q_i} p_i^q) + \sum_{q \in Q_i} h_i^q p_i^q &\leq u_i \qquad \forall i \in R \\ \sum_{q \in Q_i} (T_r + h_r^q) p_r^q &\leq u_r \qquad \forall r \in R. \end{aligned}$$

Although it is possible to include service times and impose a maximum length on truck routes with increased complexity in the formulation, we proceed for the remainder of the chapter with the assumption of zero service times and no constraints on truck route duration.

3.3.2 Path-Based Formulation

The arc-based formulation is computationally expensive, and is only appropriate for modest problem sizes. For example, Gurobi takes over one hour to find the optimal solution to a problem with one shuttle, two trucks, forty customers, and ten LMSLs. Meanwhile, it finds the optimal solution to a problem with one shuttle, two trucks, twenty customers, with ten of the customers being LMSLs within five seconds. For this reason, we develop a path-based formulation and corresponding column generation scheme, which has demonstrated success in several vehicle routing contexts. For an introduction to column generation, we refer the reader to the presentation by Barnhart et al. [61].

Let Ω be the set of all feasible shuttle routes. Let $d_r \in \mathbb{R}^+$ be the distance associated with the shuttle route $r \in \Omega$, defined as the sum of the lengths of the arcs traversed by the shuttle. Let $\delta_r^l \in \{0, 1\}$ denote whether route $r \in \Omega$ serves LMSL $l \in L$. Let the binary variable $y_r \in \{0, 1\}$ represent the decision to include the route r in the solution. The following set-partitioning formulation models the RVRP:

$$\begin{aligned} \min \sum_{r \in \Omega} & d_r y_r & & \\ \text{s.t.} \sum_{r \in \Omega} & \delta_r^l y_r &=& 1 \qquad \forall l \in L & (3.14) \\ \sum_{r \in \Omega} & y_r &=& V & (3.15) \\ & y_r &\in& \{0, 1\} \quad \forall r \in \Omega. & \end{aligned}$$

As $|\Omega|$ is, in general, exponential in the number of nodes in the network, the direct use of the formulation is intractable for problem sizes of interest, and we instead solve with column generation. Denote by MP the above (master) problem, and denote by l-MP the linearization of the MP (where the binary y_r variables are relaxed to be non-negative real). Let $\{\pi_l\}_{l \in L}$ and π^V be the dual variables associated with constraints 3.14 and 3.15 in the l-MP, respectively. The dual feasibility constraints for the l-MP are:

$$\pi^V + \sum_{l \in L} \delta_r^l \pi_l \leq d_r \qquad \forall r \in R.$$

In a column generation scheme, we instead solve a linearized reduced master problem (l-RMP), by limiting the columns to a subset of feasible routes $\Omega^0 \subset \Omega$. In our implementation, Ω^0 initially consists of two columns. The first is associated with the TSP tour through each of the LMSLs, beginning and ending at the depot. The second is associated with the solution generated by the heuristic described in Section 3.3.3. After solving the l-RMP (and in so doing, generating the dual variables $\{\pi_l\}_{l \in L}$ and π^V), we can then identify columns to add to the l-RMP by identifying the route of minimum reduced cost:

$$\min_{r\in\Omega}\left[d_r - \sum_{l\in L}\delta_r^l \pi_l - \pi^V\right]. \tag{3.16}$$

This step is referred to as the pricing problem. If the solution objective value is negative, the route is added to the l-RMP. Otherwise, the optimal solution to the l-RMP is the optimal solution to the l-MP. If the solution is binary-feasible, it provides an optimal solution to the MP. Otherwise, the solution serves as a lower bound, and we must branch-and-bound until a non-fractional optimal solution is identified. At each node in the branch-and-bound tree, we may solve the problem using the same column generation process.

With the following proposition, we note that in the one-shuttle case branching is unnecessary:

Proposition 3. *If $V = 1$, a binary-valued optimal solution can be constructed from any optimal solution to the l-MP.*

Proof. Proof. Consider an optimal solution to the l-MP, and let $Z \subseteq \{1, \ldots, |\Omega|\}$ be the indices of the nonzero y_r variables in the solution. Suppose $\delta_{r_i}^{\hat{l}} = 0$ for some $\hat{l} \in L$ and some $i \in Z$. Then,

$$\sum_{r\in\Omega}\delta_r^{\hat{l}} y_r = \sum_{i\in Z}\delta_{r_i}^{\hat{l}} y_{r_i} < \sum_{i\in Z} y_{r_i} = \sum_{r\in\Omega} y_r = 1,$$

which violates 3.14. Thus, $\delta_{r_i}^{l} = 1$ for all $i \in Z$, for all $l \in L$. Therefore, the binary-valued solution $y_{r_i} = 1$ and $y_r = 0$ for all $r \in \Omega$ such that $r \neq r_i$ is feasible for all $i \in Z$. Letting $i^* = \text{argmin}_{i\in Z} d_i$, a binary-feasible optimal solution is given by $y_{r_i^*} = 1$ and $y_r = 0$ for all $r \in \Omega$ such that $r \neq r_{i^*}$, since

$$d_{i^*} \leq \sum_{i\in Z} d_i$$

and the rightmost quantity is optimal. □

3.3.2.1 Pricing Problem

We may cast 3.16 as a resource-constrained elementary shortest path problem (RCESPP). To this end, we introduce a modified network, which can capture the distinction between visiting an LMSL on a truck route for a rendezvous and visiting the same LMSL only for a direct delivery. Without loss of generality, we assume the triangle inequality is satisfied by the road network. Let $L^0 \subseteq L$ be the set of LMSLs that can also be used as rendezvous points. For each $l \in L^0$, we create a copy at the same location which corresponds to the location only being visited for the purposes of a direct delivery. The original then corresponds to a visit to the location for a rendezvous. Denote the new, direct delivery, nodes L^*. For each LMSL with a copy, $l \in L^0 \cup L^*$, denote its copy $\hat{l}$. For each $l^* \in L^*$, let $\pi_{l^*} = \pi_{\hat{l^*}}$. We introduce the set $N^* = (N, L^*)$, which is the extension of N by appending the members of L^*. For each $i \in N^* \backslash R$, set $S_i = \emptyset$. For each $i \in N^*$, define the time of shuttle arrival limits:

$$b_i = \begin{cases} T_i & i \in N \\ \infty & i \notin N, \end{cases}$$

and for each $i \in N^*$, define the 'prize' for a shuttle making the visit to i:

$$\lambda_i = \begin{cases} \pi_i + \sum_{k \in S_i} \pi_k & i \in L^0 \cup L^* \\ \sum_{k \in S_i} \pi_k & i \notin L^0 \cup L^* \end{cases}$$

Consider the complete network $G = (N^*, E)$. Let F be the set of arcs connecting direct delivery nodes with earlier rendezvous nodes on the same truck route, as well as the set of arcs connecting rendezvous nodes with all nodes later on the same truck route. By the triangle inequality, such arcs are unnecessary, and, furthermore, deleting such arcs avoids issues with double counting prizes. Define for each arc $(i,j) \in E \backslash F$, the cost

$$e_{ij} = \begin{cases} d_{ij} - \lambda_j & j \in N^* \backslash \{n+1\} \\ d_{ij} - \pi^V & j = n+1. \end{cases}$$

The solution to 3.16 is the path through the network $G' = (N^*, E\backslash F)$, starting at $s = 0$ and ending at $t = n+1$, of minimum cost (defined as the sum of the edge costs along the path), such that, for each stop in the path, i, the arrival happens prior to the time limit, b_i, and such that each LMSL is served at most once.

To solve this RCESPP, we employ a labeling algorithm, which dynamically attributes labels to each node in the network, where a label corresponds to a path starting at s (see Ahuja et al. [62] for an introduction to labeling algorithms). Each path is associated with the resource consumption (in our case, time, t) at the path's end, as well as the accumulated cost, c.

To prevent cycles, we introduce a dummy resource for each node in N^*, with only one unit available for each dummy resource, which is consumed when the corresponding vertex is visited. Furthermore, the visit to node i consumes the dummy resource for each node in S_i. If i has a copy, the resource associated with its copy, $\hat{i}$, is also consumed. The consumption of the N^* dummy resources is indicated by a $N^* \times 1$ vector D initialized at $\mathbf{0}$. Altogether, each label has the form (D, t, c, i). Furthermore, to recover the path associated with a label, each label has a pointer to the previous label. Let D_l, t_l, c_l, and i_l denote the dummy vector, time consumption, accumulated cost, and current node associated with label l, respectively.

The algorithm is as follows. We define the set U of unextended labels and P of extended labels. Starting with $U = \{(\mathbf{0}, 0, 0, 0\}\}$, choose a member l of U, and generate new labels by 'extending' l. The extension of label l is as follows. For each $j \in N^*$ such that $D_l^j = 0, t_l + d_{i_l,j} \leq b_j$, and $(i_l, j) \in E \backslash F$ (i.e., the extension is feasible), create the new label $(D_{l'}, t_{l'}, c_{l'}, j)$, where:

$$
\begin{aligned}
D_{l'}^k &= \begin{cases} 1 & D_l^k = 1 \text{ or } k = j \text{ or } k \in S_j \text{ or } (j \in L^0 \cup L^* \text{ and } k = \hat{j}) \\ 0 & \text{otherwise} \end{cases} && \forall k \in N^* \\
t_{l'} &= \begin{cases} b_j & b_j < \infty \\ t_l + d_{i_l,j} & b_j = \infty \end{cases} \\
c_{l'} &= c_l + e_{i_l,j}.
\end{aligned}
$$

Add the new labels (if any) to U, and add l to P. Repeat until U is empty. The members of P corresponding to a complete path (i.e., those that terminate at the depot) are searched for that of minimum cost. This route is the optimal solution of 3.16. All paths of negative reduced cost (if any) are added to the l-RMP. If no path of negative reduced cost has been found, the existing solution to the l-RMP is optimal.

The run time of the labeling algorithm is largely determined by the number of labels produced. It is, therefore, desirable to remove unnecessary labels. A common technique is to test whether a newly produced label l is dominated by any label already in U or P prior to allowing it to join U. If this is the case, the label may be discarded. Furthermore, if a label dominates a label in U or P, the latter may be discarded. A label, a, dominates another label, b, only if

$$D_a^j \leq D_b^j \quad \forall j \in N^*, i_a = i_b, t_a \leq t_b, \text{ and } c_a \leq c_b, \tag{3.17}$$

and at least one of the inequalities is strict. That is, if the resource consumption and the cost of the partial path associated with a do not exceed those of the partial path associated with b, and both partial paths terminate at the same node. This dominance criterion ensures the optimal path is still found. We can also accelerate the algorithm by identifying nodes which cannot be visited in any feasible extension of a given state (i.e., unreachable nodes). When a node is found to be unreachable from a given state, the consumption of the corresponding dummy resource in that state is set to 1. This leads to more label domination, and, thus, fewer labels generated.

The choice of which label in U to extend can have a significant influence on the run time. In our implementation, labels are explored according to the nodes they are associated with. All nodes are visited in a cycle, and for each node, the algorithm extends all labels associated with the nodes that have not yet been extended. Labels associated with the same node are extended by increasing order of time consumption.

We also introduce a heuristic version of the dominance criterion, which leads to fewer labels produced:

$$\mathcal{V}_a \geq \mathcal{V}_b, i_a = i_b, t_a \leq t_b, \text{ and } c_a \leq c_b, \tag{3.18}$$

and at least one of the inequalities is strict, where $\mathcal{V}_l$ is the number of LMSLs served by the shuttle at the current state along the path associated with l. The use of this dominance criterion speeds up the pricing step, but the resulting solution may not be optimal. Once the pricing step is unable to produce a path of negative reduced cost, optimality is verified (or refuted) by a second run, using the original dominance criterion 3.17.

3.3.3 Heuristic

We now describe the heuristic developed for the one-shuttle problem. A good heuristic for the one-shuttle case can be used to find a solution for the multiple shuttle case. For example, a reasonable strategy might be to partition trucks among a fleet of shuttles, transforming a multi-shuttle problem into several one-shuttle problems.

Our heuristic is divided into two stages. The first identifies a set of candidate solutions. The second attempts to make local improvements to each of the candidates. The design of the first stage is governed by two trends we observed in optimal solutions to small test instances. First, truck-shuttle meetings (rendezvous) tend to happen early in the shuttle's route. Second, given multiple locations to rendezvous, the location that minimizes the time until the truck and shuttle meet tends to be selected. Note that the second point implies the shuttle may travel farther than necessary to meet with the truck. However, by meeting earlier with the truck, it is often the case that the shuttle is enabled to make other rendezvous before the other trucks are too far away, or have already passed their respective LMSLs.

With these trends in mind, our candidate solutions are constructed as follows. Every ordered pair of two trucks is considered. For a given ordered pair of trucks, two paths are constructed. Both paths begin at the depot, then visit the first truck as early as possible, and then visit the second truck as early as possible (starting at the rendezvous point with the first truck). For the first path, the LMSLs which are unserved by either of the two rendezvous are visited in the

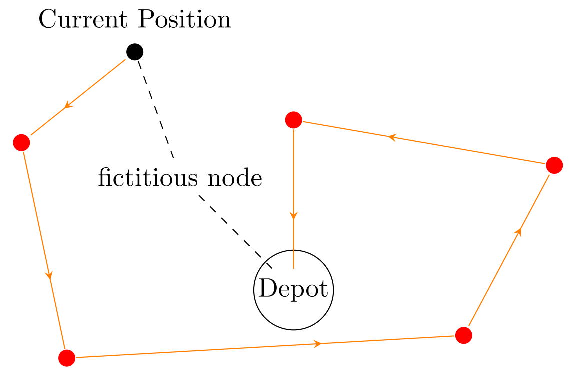


Figure 3.3: Finding the optimal shuttle return path by solving a TSP.

distance-minimizing sequence that begins at the current position (the rendezvous point with the second truck) and terminates at the depot. This last sequence is determined by solving a (typically small) Traveling Salesman Problem (TSP), using each of the remaining LMSL locations, the current shuttle location, the depot, and a fictitious node connected only to the depot and the current shuttle location. We illustrate this method in Figure 3.3. A fictitious node is included in the network, only connected to the node corresponding to the current position of the shuttle and the depot. Both arcs have a cost of 0. The TSP is solved on this network. The subpath connecting the depot, fictitious node, and the current position will necessarily be a component of the optimal tour, and the optimal shuttle path can be recovered from the solution. The orange route shows the resulting shuttle path, first visiting all the remaining LMSLs directly and then arriving at the depot. The TSP problem is solved by the LKH2 solver, a powerful open-source implementation of the Lin-Kernighan heuristic [63]. We do not have a guarantee of optimality by using this heuristic for this subproblem. However, for the problems we solve in our tests, in which $N \leq 100$, the LKH2 heuristic typically finds the optimal TSP tour.

The second path is completed using a greedy heuristic. Given the current location, we identify a set of candidate locations, which include rendezvous as well as direct deliveries. A

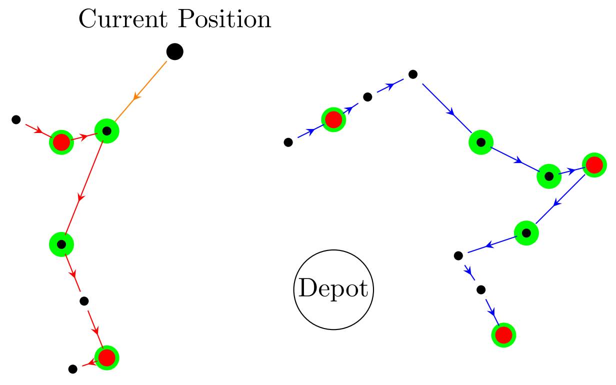


Figure 3.4: Finding a shuttle return path in a greedy fashion.

node $n \in N^*$ is a candidate if the visit would serve a yet-unserved LMSL (directly or indirectly), and, if n is a rendezvous point, the shuttle would arrive no earlier than E units of time before the truck, and no later than the truck. This prevents the shuttle from selecting rendezvous locations at which it would need wait excessively long. The nearest candidate is selected, and the heuristic repeats from the new location until all LMSLs are served, at which point the shuttle returns to the depot. See Figure 3.4 for a visualization of this procedure. In the Figure, candidate nodes are highlighted in green. All unvisited LMSLs (red) are candidates, as well as all rendezvous points (black) that the shuttle can reach from its current position within E units of time before the truck arrives. Of these, the shuttle will move to the closest to its current position, and then the process will repeat.

By design, each ordered pair of trucks gives rise to two shuttle paths (one completed by the TSP tour, the other completed greedily). In particular, if the problem includes K trucks, $2K(K-1)$ paths are produced in stage 1. We choose the best M of these paths as candidate solutions, which are passed to the second stage for local improvement.

We now describe the second stage of our heuristic, which attempts to improve upon solutions generated by the first stage. Starting with a solution produced by the first stage, we apply six different improvement procedures sequentially, and repeat until the most recent pass of the procedures fails to improve upon the existing solution. To aid the description of these procedures, we introduce the following notation. A shuttle route is denoted $r = \{s_1, s_2, \ldots, s_m\}$, where m is the total number of stops on the route, and $s_i \in N^*$. Note that s_1 and s_m correspond to the depot at the beginning and end, respectively. Let τ_i denote the time at which service at stop s_i is completed. We now describe each of the six procedures, in the order in which they are applied:

3-Opt: We consider the modifications of r that arise from deleting three connections, reversing the order of some of the resulting subpaths, and reconnecting. Specifically, for each $i, j, k \in \{2, \ldots, m-1\}$ such that $i < j < k$, we consider each reconnection of the paths: $\{s_1\}$, $\{s_2, \ldots, s_i\}$, $\{s_{i+1}, \ldots, s_j\}$, $\{s_{j+1}, \ldots, s_k\}$, and $\{s_{k+1}, \ldots, s_m\}$, where we may reverse any of the middle three paths (a total of seven possibilities). If, for given i, j, k, one of the options forms a feasible route of improved objective value, r is replaced with the route, and the process repeats from the beginning. If no such route exists for all i, j, k, the process terminates.

Add-Rendezvous: We proceed, in order of visits, through the route r. If stop s_i is a direct delivery (i.e., if $b_{s_i} = \infty$), we consider inserting a rendezvous into the route that serves the corresponding LMSL. Denote the rendezvous nodes considered $Q_{s_i} \subseteq N^*$. A node $j \in N^*$ is in Q_{s_i} if $s_i \in S_j$ and $\tau_{i-1} + d_{i-1,j} \leq b_j$. That is, a rendezvous at j is considered if it serves s_i, and if it is possible to intercept the truck at j after the visit to s_{i-1}. Given the rendezvous $j \in Q_{r_i}$, we consider the route created by deleting s_i from r (as well as any other stop in r made unnecessary by a visit to node j) and inserting j into r in the cheapest feasible way (without changing the order of the other visits). If such a route exists and decreases the objective cost, the solution is replaced and the procedure repeats.

Remove-Rendezvous: We proceed, in order of visits, through the route r. If stop s_i is a rendezvous point (i.e., if $b_{s_i} < \infty$ and $i \neq 1$), and is such that the visit to s_i indirectly serves at most two LMSLs yet unserved by the route, we consider removing the rendezvous s_i. This is done by examining the (at most two) routes that arise from replacing s_i with the direct deliveries while preserving the order of the other stops in r. The shortest valid route is selected if the total distance is at most five percent greater than the original distance.

The procedure allows 'uphill' moves, and is motivated by the opportunity cost of a shuttle waiting a long time for a truck, as other rendezvous become impossible. By removing such rendezvous, the shuttle then has greater flexibility to meet with other trucks later on.

Move-Rendezvous: We proceed, in order of visits, through the route r. If stop s_i is a rendezvous point (i.e., if $b_{s_i} < \infty$ and $i \neq 1$), each member of the set of equivalent rendezvous points, $P_{s_i} \subseteq N^*$, is considered to replace s_i. A node $j \in N^*$ is in P_{s_i} if $S_{s_i} \subseteq S_j$, and the route resulting from replacing s_i with j is feasible. We then identify the replacement that minimizes the total distance, while keeping the arrival time at r_{i+1} within 10% of the original time of arrival, if it exists. If such a replacement exists, and the resulting route improves the objective value, the solution is replaced and the procedure repeats.

Add-Rendezvous (version 2): We run this procedure again, but with a different criterion for possible rendezvous nodes, Q_{s_i}. A node $j \in N^*$ is in Q_{s_i} if $s_i \in S_j$ and $b_j - E \leq \tau_{i-1} + d_{i-1,j} \leq b_j$. That is, we consider adding those rendezvous points that serve the LMSL at s_i that can be reached after the visit to the previous node, s_i, but not earlier than E units of time before the truck arrives. The latter constraint removes rendezvous points which lead to long idle times for the shuttle.

Reinsert: We proceed, in order of visits, through route r. For each stop $r_i \in r$, and for each $j \in \{2, \ldots, m\}$, we evaluate the modified route, r^*, which moves r_i from its current position to directly in front of r_j. Furthermore, the path r^* is passed through the Move-Rendezvous procedure, with the restriction on arrival times relaxed. If r^* is feasible and improves the objective value, r^* replaces r and the process repeats.

3.4 Computational Results

We now conduct a computational study for the RVRP. We first describe the data sets we developed for the RVRP, and evaluate the relative performances of the three approaches. We then perform a sensitivity analysis of the utility of truck-shuttle synchronization against problem parameter settings.

3.4.1 Test Data

We now describe procedure used to construct our test instances. Coordinates of service locations and the depot were obtained from the well-known Solomon data set [64]. This data set provides random, clustered, and mixed location data for 25, 50, and 100-customer instances (altogether nine instances). We use the Euclidean metric for distances. The Solomon data provides [x,y] integer-valued coordinates drawn from a $[0, 100] \times [0, 100]$ grid. With this data, we design the fixed truck routes. The set of customers is partitioned among the trucks by a k-means clustering algorithm (where k is set to the number of trucks, T). Each truck is then sequenced through its cluster with the LKH2 solver [63]. Then, given a uniform 'head start' parameter, $h \in \mathbb{R}$, each of the trucks is initialized at the first location on their route visited after h units of time have elapsed since departure from the depot (where travel times are equal to distances). This is their location at the beginning of the problem. Finally, we randomly choose $l\%$ of the service locations to be LMSLs (rounding up to the nearest integer). In our initial tests, we set $T = 4$, $h = 10$, and $l = 10$. Any changes to these nominal values for the purpose of testing will be made explicit.

In order to evaluate the approaches on larger instances, we applied the same process on the Gehring and Homberger data set [65]. This data set provides random, clustered, and mixed location data for instances with 200, 400, 600, 800, and 1000 customers. Locations are drawn from a $[0, 150] \times [0, 150]$ grid for the 200-customer instances, and a $[0, N/2] \times [0, N/2]$ grid for the rest, where N is the number of customers. For these instances, we set $T = 4$, $h = 10$, and $l = 5$. The reason l is decreased for these instances is that this lower value is more realistic in practice (i.e., only 5% of consumers may have ordered packages for same day delivery). The parameter is kept at 10 for the Solomon instances (which are smaller) to keep the problems more challenging.

We denote the instances with randomly, clustered, and mixed location data by R, C, and RC, respectively, concatenated with the number of customers (25, 50, and 100 from the Solomon set, and 200, 400, 600, 800, 1000 from the Gehring and Homberger set). For example, the instance with clustered location data and 400 customers is drawn from the Gehring and Homberger data set, and is denoted $C400$.

We make a couple of notes on implementation. First, in Section 3.3.2, the RVRP is cast as a set-partitioning problem. However, we found it slightly computationally advantageous to relax the master problem into a set-covering problem. Specifically, we allow the left hand side in Constraint 3.14 to be greater than the right hand side. This only affects our procedure in that it restricts the dual variables to non-negative values. Second, in the heuristic presented in Section 3.3.3, we must specify two parameters: M, the number of paths produced in stage 1 to pass to stage 2, and E, the limit on how early a shuttle can arrive before a truck at a given node. After some brief experimentation, we found $M = 10$ and $E = 50$ find a good balance between solution quality and efficiency. Our results reported for the heuristic were found with these settings. All run times reported are in seconds. The data sets generated for this study are made available at `https://github.com/ericoden/RVRP-data`.

3.4.2 Comparison of Approaches

We investigated the relative performances of the arc-based and path-based methods, described in Sections 3.3.1 and 3.3.2, respectively. In order for the path-based approach to be exact, column generation needs to be embedded in a branching procedure in the event of fractional solutions. However, in the one-shuttle case, this is unnecessary, as an integer optimal solution always exists to the LP relaxation if the reduced master problem is feasible. Thus, for one shuttle problems, we may compare the two approaches in terms of their ability to identify and prove optimality. We present the performances on the problems based on the Solomon data set in Table 3.1.

For all nine instances, the path-based approach identifies and proves the optimal solution within 11 seconds. Meanwhile, the arc-based approach is only able to do this with the 25-customer instances within a five minute time limit. In the rightmost columns, we record the optimality gap of the incumbent solution of the arc-based method (relative to the true optimal solution identified by the path-based method), as well as the current MIP gap (the gap relative to the MIP lower bound), after one minute and after five minutes. These columns indicate that the arc-based approach is able to quickly identify the best solution for the 50-customer instances, but

	Time to Prove Optimality (sec)		Gaps for Arc-Based (%)			
Instance	Arc-Based	Path-Based	1 Min		5 Min	
			True	MIP	True	MIP
R25	0.23	1.08	0	0	0	0
R50	-	0.80	0	42	0	29
R100	-	10.61	23	67	1	54
C25	0.06	0.61	0	0	0	0
C50	-	0.69	0	39	0	35
C100	-	4.14	19	87	19	84
RC25	0.57	0.41	0	0	0	0
RC50	-	0.67	0	54	0	50
RC100	-	9.40	16	68	16	61

Table 3.1: Performance of the arc-based and path-based methods.

struggles to prove optimality. Altogether, these experiments confirm the computational superiority of the path-based approach in this context.

However, the path-based approach struggles on the larger instances derived from the Gehring and Homberger data set. In Table 3.2, the objective values and run times (with a one hour cap) are reported. A dash indicates the method failed to terminate within an hour, and the corresponding objective value is that of the incumbent solution. We observe the exact algorithm is only able to produce and verify the optimal solution for the 200 customer instances within the time limit. However, the heuristic presented in Section 3.3.3 performs quite well on these instances, either matching or exceeding the solutions from the path-based method. For these instances, we also present the (optimal or near optimal) objective value of the tour which makes only direct deliveries, which we denote by TSP. These values were found by the LKH2 solver. We observe the significant savings associated with allowing truck rendezvous (e.g., 77% in the R1000 instance). Next, the experiments in Table 3.2 are repeated, with the single change of increasing the number of trucks to eight. We observe similar results, reported in Table 3.3.

Instance	TSP	Path-Based		Heuristic	
		Obj	Time	Obj	Time
R200	416.76	287.55	50.10	287.55	1.15
R400	858.35	447.58	-	447.58	3.72
R600	1435.24	378.49	-	371.76	4.63
R800	2141.68	973.36	-	754.67	14.02
R1000	2995.27	910.18	-	674.21	9.87
C200	448.90	356.26	42.61	356.26	0.97
C400	599.28	501.73	-	501.73	8.13
C600	1233.63	830.07	-	680.15	8.07
C800	1967.97	1326.43	-	1098.79	47.01
C1000	2810.40	1752.25	-	1428.86	43.45
RC200	477.89	351.85	91.87	351.85	1.32
RC400	778.16	409.26	-	409.26	2.77
RC600	1250.13	782.87	-	532.04	9.87
RC800	2019.16	1357.22	-	838.73	29.38
RC1000	2746.30	1421.92	-	983.70	21.76

Table 3.2: Performance of the methods on larger instances of the RVRP.

The path-based method is still capable of quickly identifying the optimal solution for the 200 customer instances, but cannot for any of the larger instances. Meanwhile, the heuristic solutions either match or improve upon the incumbent solutions produced by the path-based method after an hour of computation.

To verify the effectiveness of our heuristic, we constructed one hundred instances of R200, where the locations of the LMSLs were randomized in each instance. Furthermore, the seeds for the k-means clustering algorithm were also randomized, giving rise to different truck routes among the instances. We then compared the solution produced by the heuristic with the optimal solution, which can be produced relatively quickly for such instances using the path-based method. The histogram in Figure 3.5 presents the number of instances whose heuristic solution match the optimal solution in objective value, and the number of instances where the percent optimality gap falls within the intervals $(0, 1]$, $(1, 2]$, $(2, 3]$, $(3, 4]$, $(4, 5]$, and $(5, \infty)$. The same experiment

Instance	TSP	Path-Based		Heuristic	
		Obj	Time	Obj	Time
R200	416.76	327.17	5.34	327.17	2.76
R400	858.35	759.88	-	746.24	11.55
R600	1435.24	1434.54	-	1247.50	32.05
R800	2141.68	2070.17	-	1656.33	76.76
R1000	2995.27	2842.98	-	2300.17	167.90
C200	448.90	433.50	8.24	433.50	2.96
C400	599.28	552.74	-	552.87	14.42
C600	1233.63	1116.12	-	1070.44	32.28
C800	1967.97	1967.97	-	1855.85	134.39
C1000	2810.40	2801.28	-	2549.73	274.85
RC200	477.89	450.91	27.40	450.91	3.43
RC400	778.16	637.38	-	624.78	10.48
RC600	1250.13	1130.23	-	1075.99	43.85
RC800	2019.16	1966.30	-	1697.60	88.79
RC1000	2746.30	2591.95	-	2456.85	234.20

Table 3.3: Performance of the methods, with the number of trucks raised to eight.

was carried out with the number of trucks set to eight. We observe the heuristic is quite effective, matching the exact solution around 80% of the time, and on average below 1% optimality gaps for both four and eight truck instances. Furthermore, in neither of the sets of instances does the gap between the heuristic and optimal solution exceed 5%. We notice the heuristic performs slightly better on instances with eight trucks. A possible explanation is that there are fewer LMSLs per truck in this case, and, thus, the set of highly distinct solutions is smaller. For instance, if a truck has one LMSL on the route, a shuttle can either rendezvous with the truck beforehand, or make the direct delivery. If the truck has two LMSLs, the shuttle can rendezvous before the first, rendezvous between the first and the second (directly delivering to the first), or directly deliver both.

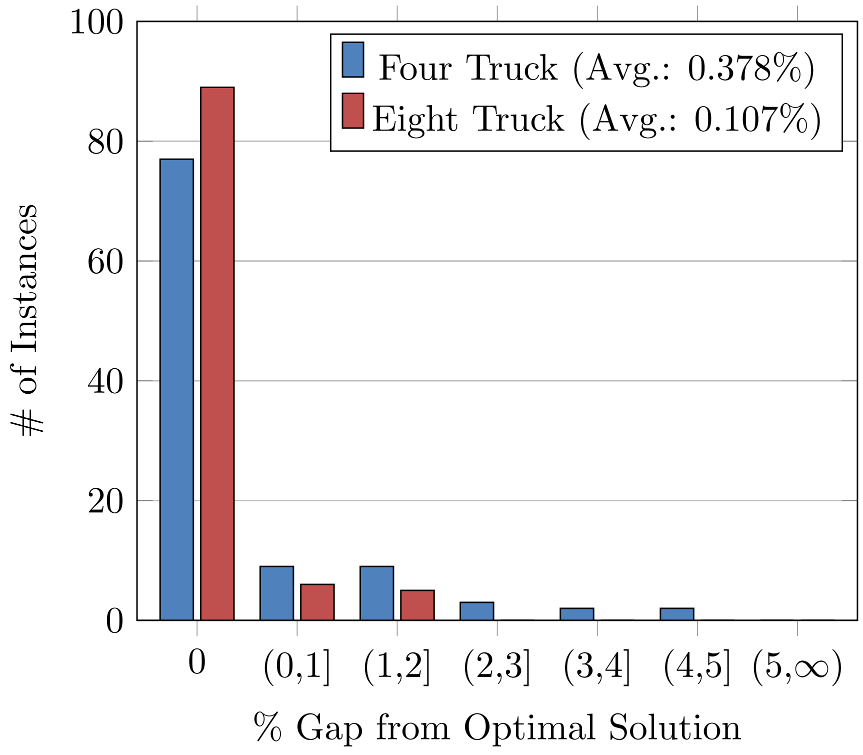


Figure 3.5: Performance of heuristic on several 200-customer instances.

3.4.3 Input Sensitivity

Among our goals in this work is to characterize when a shuttle-truck synchronization policy works to the greatest benefit. We, therefore, are interested in comparing the optimal use of such a policy against the optimal solution with only direct deliveries (i.e., the TSP solution). We focus on the smaller instances (i.e., the Solomon instances and the Gehring and Homberger instances where the number of stops is 200), where our path-based method can generate optimal solutions within a reasonable length of time.

In Table 3.4, we compare the TSP solution with shuttle-truck synchronization (i.e., allowing indirect deliveries). We test the latter with both one shuttle and two shuttles available using our exact path-based approach. For each of the instances, the solution at the root node for our column generation procedure was optimal, meaning branching was unnecessary (this is not guaranteed to happen in the multiple shuttle case).

In four smaller instances (R25, C25, RC25, and RC50), there is little benefit from allowing rendezvous; the optimal traveling salesman tour through the LMSLs is the optimal (or nearly optimal) solution to the RVRP. However, as the instances become larger, the benefit of allowing rendezvous emerges. We also remark that instances in which locations are randomly determined are the most conducive to synchronization, and that the clustered locations are the least conducive. This might be explained by the resulting proximity of truck routes to one another in the former case, enabling a shuttle to quickly travel from one rendezvous to another. We finally notice that in the smaller instances, there is little benefit to a second shuttle. In all but one of the instances with 50 customers or fewer, the optimal solution only uses a single shuttle. This is largely due to the objective of minimized distance. Situations in which alternative objectives (such as minimization of the time before the shuttles return) are more relevant could benefit from a larger fleet of shuttles.

We then investigated how LMSL density (that is, the proportion of truck stops that are LMSLs) affects the utility of shuttle-truck synchronization. The instances from Table 3.4 were modified by doubling the number of LMSLs for each instance. Table 3.5 reports the results. We observe the TSP solution is unsurprisingly increased, and we also observe the improvement using shuttles is similar to the results in the previous experiment, with the larger instances seeing more improvement. The extent of the improvement is about the same, indicating that it is the availability of rendezvous locations, rather than the number of LMSLs, that drives the performance improvement.

It is easy to see that the position of the trucks at the beginning of the problem influence

Instance	TSP	RVRP (1 shuttle)		RVRP (2 shuttles)	
	Obj	Obj	Savings (%)	Obj	Savings (%)
R25	116.92	116.55	0.32	116.55	0.32
R50	189.23	172.56	8.81	172.56	8.81
R100	205.98	152.10	26.16	152.10	26.16
R200	416.76	287.55	31.00	249.99	40.02
C25	87.68	87.68	0 .00	87.68	0.00
C50	103.77	97.37	6.17	97.37	6.17
C100	202.71	171.88	15.21	169.16	16.55
C200	448.90	356.26	20.64	290.29	35.33
RC25	133.06	133.06	0.00	133.06	0.00
RC50	262.84	262.84	0.00	262.84	0.00
RC100	245.21	194.00	20.88	194.00	20.88
RC200	477.89	351.85	26.37	294.98	38.27

Table 3.4: Without allowing vs. allowing rendezvous

the efficiency of synchronization. The further away they are at the outset, the further the shuttle must travel to intercept them, and these effects compound if a shuttle attempts to make multiple rendezvous. We conducted an experiment to observe this effect. We created 100 instances of the problem, with the location data given by the R50 data. Problems differed only by which of the stops were set to be LMSLs. Then, for each instance, and for each setting of the truck head start parameter h in $\{5, 10, 15, \dots, 100\}$, the optimal solutions for one and two shuttles are determined. The average improvements upon the TSP solution are shown in Figure 3.6. We see, as expected, a sharp decrease in utility as the head start of the trucks increases, eventually leveling off towards the same value as the trivial solution. This behavior is explained by the observation as h is increased, trucks are more likely to have passed their LMSLs, and thus, there are fewer opportunities for synchronization. In practice, the synchronization approach could yield significant benefits when a large fraction of the truck routes remain to be completed.

Instance	TSP	RVRP (1 shuttle)		RVRP (2 shuttles)	
	Obj	Obj	Savings (%)	Obj	Savings (%)
R25	156.98	156.25	0.47	156.25	0.47
R50	212.52	198.70	6.50	198.70	6.50
R100	271.12	169.55	37.46	156.22	42.38
R200	492.42	356.37	27.62	338.48	31.26
C25	97.87	97.87	0.00	97.87	0.00
C50	136.16	125.40	7.90	125.40	7.90
C100	308.47	265.05	14.08	247.24	19.85
C200	578.44	481.47	16.76	427.59	26.07
RC25	164.62	164.62	0.00	164.62	0.00
RC50	270.88	270.88	0.00	270.88	0.00
RC100	355.41	292.11	17.81	272.33	23.38
RC200	518.93	379.30	26.91	331.72	36.08

Table 3.5: Without allowing vs. allowing rendezvous, number of LMSLs doubled.

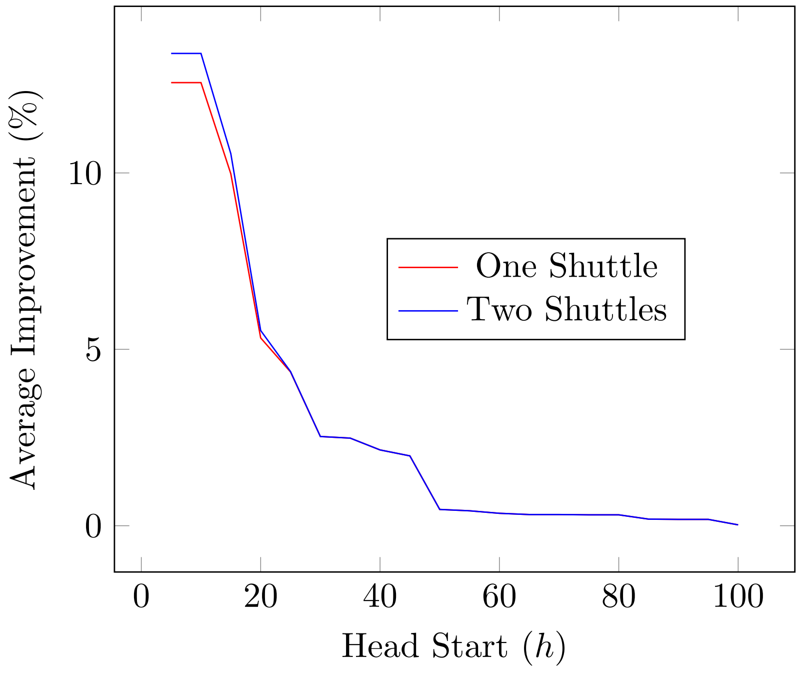


Figure 3.6: Sensitivity analysis of the head start parameter, h.

3.5 Conclusions

In this chapter, we presented a new vehicle routing problem with relevance to contemporary last-minute shipping concerns. We propose a column generation approach as a quick and effective means to generate optimal routes for problem instances with up to 200 nodes. Furthermore, we propose our two-stage heuristic as a reliable means of generating solutions for larger problem instances, with up to 1000 nodes. Our computational experiments demonstrate that allowing shuttles to rendezvous with trucks mid-route can save up to 77% on travel costs.

Our computational experiments illustrated situations appropriate for employing truck-shuttle synchronization. In particular, the larger the instances, the greater the benefit. If trucks have more stops along their routes, shuttles have more opportunities to meet with them. Truck-shuttle rendezvous are most suitable in environments where truck routes are fairly close to each other, as a shuttle route can leverage this proximity to minimize travel distance.

There are several possibilities for further research. One natural question to explore is: How do the results vary as a function of the 'head start' parameter h? This parameter defines a batching strategy for LMSLs. If h is too small, the objective value and the number of LMSLs serviced will be small. If h is too large, the objective value and the number of LMSLs serviced will be much larger. The challenge is to determine a smart batch size h. In this chapter, we assumed that h was already known.

A second extension deals with designing truck routes based on a future (probabilistic) forecast of same-day demand. The idea being that accounting for same-day demand (along with scheduled package delivery) in designing truck routes, may yield additional savings opportunities. This could be cast as a bilevel program that minimizes expected total travel costs (i.e., for both trucks and shuttles) subject to the design of truck routes, where shuttles are routed optimally given the design of the truck routes. Another extension deals with allowing for multiple shuttle dispatches throughout the day, making the problem much more dynamic. Questions that arise in this setting include when to (optimally) dispatch shuttles, and how to determine an appropriate rendezvous/delivery strategy. We believe the benefits of shuttle-truck synchronization would increase in such a setting, and, therefore, we intend to continue our work in this direction.

Chapter 4: The Urban Air Mobility Problem

4.1 Introduction

Traffic congestion has become one of the greatest detriments to urban living. A key challenge is to produce clean, scalable, cost-efficient technologies to improve mobility in cities of the future. Looking ahead, many aerospace companies have focused on electric vertical takeoff and landing (eVTOL) aircraft that can provide safe, rapid transport in an urban environment without the need of significant additional infrastructure (i.e., no need for a runway, etc.) as a technology to address this problem. Some estimates predict a reduction of an average of three hours of daily travel time for a commuter in New Delhi [66] when opting to fly rather than drive. This burgeoning area of research, encompassing a large variety of logistical, engineering, and regulatory concerns, is referred to by the umbrella term Urban Air Mobility (UAM), or 'Flying Taxis' by the popular press.

A form of UAM service already exists in the form of helicopter flights [67], which are used, for example, to bypass traffic on time-sensitive trips to the airport. As the service exists currently, it is considered a luxury service rather than a regular commuting option. However, the promise of eVTOL aircraft, with their order of magnitude advantages on noise, manufacturing cost, and fuel efficiency, has resulted in a belief that such technology can deliver as a mass market option for rapid intracity travel. Indeed, many aerospace companies have ventured into this industry. Several manufacturers, including Airbus, Airspace Experience Technologies, Aurora Flight Sciences, Bell Helicopter, The Boeing Company, EHANG, Embraer, Karem Aircraft, Kitty Hawk, Lilium [they already have a prototype capable of carrying five passengers, achieving speeds of 186 miles per hour and landing in narrow urban areas, see 68], Neva Aerospace, Opener, Pipistrel, Volocopter, and Workhorse Group are seeking eVTOL certification.

The white papers put out by the eVTOL industry [see 66, 69, 70] envision the market to be largely an airborne taxi service, integrated with a ridesharing platform. The eVTOLs will operate out of specially designed transportation hubs referred to as 'ports'. Presently, the industry is largely focused on regulatory and engineering aspects in making eVTOLS a reality. While these are key in the development of the technology, network logistics associated with the operation of the UAM system (once certified and commercialized) are likely to be another key factor in the success of this market. Particular challenges operators face include the temporal nature of demand, time windows for customers, and battery management constraints for the eVTOLs.

In this chapter, we take the perspective of a company in the early stages of development of an urban air transportation network. In particular, we assume that demand exceeds supply, due to a limited fleet size. This represents a divergence from many vehicle routing applications, which assume a steady-state environment, where enough resources exist to accommodate all requests. In the more typical setting, the objective is usually to minimize operational costs. In contrast, the priority of a new company (using a new technology) intending to rapidly grow its service may be to maximize market share. Indeed, the importance of this objective was emphasized by our industry contact. For the UAM service provider, this can mean maximizing the throughput of their service, i.e., getting as many passengers aboard their eVTOLs as possible.

The rest of this chapter is organized as follows. Section 4.2 provides relevant background, modeling assumptions, and a description of the problem studied in this chapter. Section 4.3 discusses relevant literature. Section 4.4 develops an arc-based mixed integer program formulation of the problem. However, this approach is not scalable and has computational difficulties dealing with battery management constraints. Consequently, Section 4.5 develops a path-based approach that can adequately address the scale of problems anticipated in practice (at least, initially). First, Section 4.5.1 introduces the path-based formulation, while Sections 4.5.2 and 4.5.3 describe the associated column generation procedure and pricing problem (which is a shortest path problem with resource constraints), respectively. Section 4.5.4 provides details on the initialization and Section 4.5.5 explains our branching scheme. Section 4.6 provides a detailed computational study, which includes a case study involving the Washington D.C. metropolitan area that demonstrates the ability of our approach to solve (anticipated) real-world instances. Section 4.7 discusses future areas of research in the UAM arena.

4.2 Problem Background

Our knowledge of the network logistics problems has been gained by industry white papers [66, 69, 70], participation in industry sponsored conferences (e.g., Uber Elevate Summit 2019, Washington D.C.), and contacts in the industry. Based on these, network logistics for the UAM provider (in the early phases) can be separated into three distinct and interrelated problems.

First, where should eVTOL ports be located? The success of this service will largely be dictated by the convenience of the ports for the users. Therefore, ports will have to be constructed to accommodate the expected travel demand (both existing and latent). It is expected that the eVTOLs will be used as one leg in a multi-modal transportation trip. Thus, passengers are likely to weigh the tradeoffs of a direct road trip against a multi-modal trip with up to three legs (road trip to eVTOL port, eVTOL trip to destination port, road trip from eVTOL port to destination). Second, a daily schedule for the eVTOLs must be set. Although it is envisaged that UAM service at maturity will be a fully on-demand service, it is expected that in the multi-modal ridesharing setting where the service will be deployed, to reduce complexities of the system, the eVTOL schedule will be fixed (while the road legs will be scheduled dynamically when a request comes in), customers will be offered an eVTOL option if there is capacity available on their trip. Thus, given the location of ports and daily (potential) trip demands (i.e., origin port, destination port, and time window at origin) the goal is to develop an *a priori* eVTOL schedule to maximize customer throughput. Third, since day-to-day demand patterns may vary, when required (e.g., if it becomes clear that changing the schedule will result in a significant improvement in throughput), the provider should be able to modify the *a priori* schedule in real time to improve throughput.

In this chapter, we concentrate on the second problem (determining an *a priori* schedule for the eVTOLs based on the anticipated demand) for several reasons. The placement of ports is a one time decision as they are capital intensive and require several years of planning and construction (currently UAM system providers are planning on retrofitting roofs of large garages as ports). Although their location could be optimized, there are other factors that play a larger role in determining their location (e.g., tax incentives, partnerships with garage operators, etc). Once port locations are fixed, the success of the UAM system depends on its ability to route as much demand as possible. Thus, the *a priori* routing and scheduling problem is critical. Further, as demand changes and grows, it will be necessary to repeatedly solve this problem. With eVTOL service several years away from being ready for commercialization (several cities including Dallas, Los Angeles, and Melbourne (Australia) have announced plans and hope to pilot ride sharing transportation networks that use eVTOLs, potentially as early as 2023, but this could be delayed as eVTOLs continue to be designed and await certification, a lengthy process) the parameters necessary to formulate the third problem are unclear at this juncture. Finally, we note that a solution procedure for the second problem can be a key tool for strategic planning for the UAM service provider (i.e., it could be used right away as it plans ahead for service and allows one to answer many strategic what-if questions and explore scenarios). With this background, we now elaborate and explicitly describe the second problem.

Suppose a provider possesses a fleet of eVTOLs, each of a finite, uniform capacity and a finite, uniform charge capacity. There is a set of ports dispersed throughout the area at which the eVTOLs are limited to take off and land. The charge level of each eVTOL is reduced as it travels, but may be recharged when stationed (either by swapping the battery or by plugging the battery into a power source). The provider has a set of customers who wish to use the eVTOLs over some time horizon (e.g., this could be the entire day, or the day could be broken up into pieces like an AM peak, PM peak, and off-peak periods). Each customer is characterized by an origin port, destination port, and a time window in which the departure is requested. The customer may only be routed on a nonstop flight from her origin port to her destination port. Given that the eVTOL leg is one of multiple legs in a multi-modal trip, and that the eVTOLs have a limited capacity, it is expected to be more efficient operationally and profitable in terms of maximizing revenue to only allow direct legs for the eVTOL portion. Naturally, allowing only non-stop flights reduces the flexibility of the system overall. However, this constraint was confirmed by our industry contact.

At the beginning of the time horizon, the eVTOLs may be placed at any of the ports (facilitated by overnight relocation). The charge level of each of the eVTOLs is initialized to a given value (typically full charge), and must remain above the minimum charge level. An eVTOL can pick up a passenger if the eVTOL is currently located at the customer's origin port, the current time is within the passenger's time window, and the eVTOL immediately flies directly to the passenger's destination port. The number of passengers the eVTOL can carry cannot exceed the eVTOL's capacity. Each eVTOL's charge level decreases as it flies from one port to another, but can increase while the eVTOL is stationary at a port (by recharging). We seek to maximize the total number of customers served. We call this the Urban Air Mobility Problem (UAMP).

We note that the UAMP (as described) is deterministic (significant variations from the planned for demand that necessitate schedule changes would be addressed on a day-to-day basis in real-time). In this chapter, we assume a linear charge/discharge behavior for the batteries. However, nonlinear charge/discharge behavior (which is more typical as factors including payload and current charge level influence the recharge/drainage rates) is easily incorporated into the path-based approach described in Section 4.5. Note that a given customer is associated with a single origin port and a single destination port. It may be the case that in a multi-modal trip, there are multiple possibilities for the eVTOL leg, with differing origins and destinations (this could be the case in a network with a very large number of eVTOL ports). This distinction is equivalent to that between single and multiple allocation in hub network design [71]. However, in our modeling, we assume that a given customer's demand will only be associated with a single origin-destination port pair. In the early phases, the possibility of having multiple possible origin-destination pairs for the eVTOL leg is low due to the fact that there will be few ports. We note, however, with minor adaptations (as described in Appendix B.1.0.1), one can modify our model to account for multiple port options for a given customer.

4.3 Related Literature

To the best of our knowledge, there is no published work in the operations research literature on the UAMP. There are a handful of papers (all of these research efforts are largely in parallel to ours) that deal with various aspects of an urban air mobility system. Chen et al. [72], Lim and Hwang [73], Wang et al. [74], and Wu and Zhang [75] focus on the port placement problem. Specifically Chen et al. [72] describe a variable neighborhood search heuristic that is able to solve 144 node instances (of the port placement problem) rapidly. Lim and Hwang [73] used a k-means algorithm to cluster the commuting data and select ports for Seoul, Korea. Wang et al. [74] describe a sophisticated adaptive discretization approach to approximately solve (typically within 1% of optimality) a port placement problem that accounts for tradeoffs between the strategic aspects of the port placement problem, tactical aspects of UAM operations, and customer adoption. Wu and Zhang [75] build an integer programming model for the port placement problem that accounts for customer's mode choice between ground transportation and multimodal UAM service. They apply their model to a case study in the Tampa Bay (Florida, USA) area. Pelegrin and D'Ambrosio [76] and Tang et al. [77] focus on automated route planning and collision avoidance systems in UAM systems. This is an important issue since it is envisaged that eVTOLs will be "pilotless". Garrow et al. [78] and [79] focus on survey data and market segmentation analysis to estimate demand for air taxi service.

We note that the UAMP has similarities with the well-studied Dial-a-Ride Problem (DARP), first introduced in the 1970s, which seeks the minimum-cost routing of a fleet of vehicles such that each customer is picked up and dropped off at nodes within passenger-set time windows. Multiple passengers can simultaneously occupy a given vehicle, but the ride time of each customer must not exceed some maximum length. Mathematical models of the DARP are presented by Cordeau and Laporte [80] and Ropke et al. [81]. For an account of the state of recent work on the DARP, we refer the reader to the survey presented by Ho et al. [25]. According to the survey, the most successful exact method for solving the DARP is the branch-and-cut-and-price approach developed by Gschwind and Irnich [82]. The strongest heuristic method is the adaptive large neighborhood search approach created by Gschwind and Drexl [83]. While the Dial-a-Ride problem guarantees a level of service through limiting the duration of each passenger's travel, this is distinct from the guarantee of a non-stop journey from origin to destination which is assumed in this setting.

With the increasing presence of electrically powered automobiles on urban road networks, many vehicle routing problems have been adapted to incorporate battery life constraints. Energy-consumption models have been developed expressly for this purpose [e.g., 84, 85, 86]. Such models can be used to predict charging and discharging rates for use in transportation modeling. In particular, Masmoudi et al. [87] solve an electric DARP using an Evolutionary Variable Neighborhood Search approach to solve a mixed-integer program, using the model presented in Genikomsakis and Mitrentsis [84] to determine energy consumption. In the work of Masmoudi et al. [87], recharging is accomplished by battery swapping stations dispersed throughout the network. In the work of Bongiovanni et al. [88], a branch-and-cut approach is used to solve the electric DARP.

We develop a discrete-time model of the UAMP, which maps routing/scheduling decision points onto a time-expanded network. This technique for representing a problem involving temporal constraints has been applied in many contexts. For example, see Marshall et al. [89] and Bsaybes et al. [90].

There are two important differences between the UAMP and the above problems. First, rather than minimizing the routing costs while guaranteeing that each customer is served, we seek to maximize the number of customers served. This stems from our perspective of a UAM provider in early development. Rather than the 'steady-state' environment, where the priority is to minimize operational costs, this 'early-state' seeks to maximize market share. Second, in each of the earlier problems, routes need not travel to a customer's destination node immediately after traveling to the request's origin node. Instead, quality of service considerations are incorporated by including time windows on arrival times and maximum trip lengths. As stated in Section 4.2, our assumption of non-stop passenger flights is more appropriate here.

4.4 Arc-Based Approach

In our work, we have focused on a discrete-time model of the problem, where the time horizon is partitioned into time steps. We will describe both an arc-based and path-based approach in this discrete time setting. It has been noted that there is a price to pay for the quantization of time in some models [91]. However, this approximation can be justified as an operational reality. Though improved routing may be possible if flight departures could be scheduled down to the second, it is perhaps unreasonable to expect such precision in reality. If flight times are rounded up to the nearest time step (say, five minutes), we automatically incorporate temporal 'padding' for any proposed schedule. We now describe an arc-based mathematical formulation of the problem. We first define notation and then present the formulation.

4.4.1 Notation

Let N be the set of eVTOL ports, and let $T = \{0, 1, \dots, t_{\text{max}}\} \subset \mathbb{Z}$ be the discrete time horizon. Each time step corresponds to a fixed amount of time, τ. For instance, if the service is to begin at 8:00 am, and the time step, τ, is five minutes, then 0 corresponds to 8:00 am, 1 to 8:05 am, and so on. Let C be the set of customers. Each customer $c \in C$ is associated with an origin port $o_c \in N$, a destination port $d_c \in N$, and a set of times $T_c \subseteq T$ in which the customer can depart from her origin node. T_c must be a contiguous set of times. That is, if $t_c^a = \min\{t|t \in T_c\}$, $t_c^b = \max\{t|t \in T_c\}$, and $t \in T$ is such that $t_c^a \leq t \leq t_c^b$, then $t \in T_c$. Thus, T_c represents the discrete time window for customer c. Let $K = \{1, \dots, |K|\}$ denote the fleet of eVTOLs, and let $S \in Z^+$ be the capacity (number of seats) of each eVTOL. Let Q_+, Q_-, and Q_0 denote the uniform maximum, minimum, and initial charge levels of each eVTOL, respectively.

Define the complete, directed graph $G = (N, A)$. Each arc $(i, j) \in A$ is associated with a travel time, $t_{ij} \in \mathbb{Z}^+ \cup \{0\}$ (this quantity may include take-off, landing, and some minimum 'turnaround' time allocated for inter-flight processes such as deplaning, boarding, etc.). The units here correspond to time steps, as opposed to minutes. Without loss of generality, we assume travel times satisfy the triangle inequality. Define the directed, time-expanded graph $H = (M \cup \{s, f\}, B)$, where $M = N \times T$, s and f represent fictitious origin and destination nodes, respectively, and B is the set of directed arcs connecting members of $M \cup \{s, f\}$. Each node $i \in M$ corresponds to a port-time pair (p, t), where $p \in N$ and $t \in T$. When the port associated with node $i \in M$ must be specified, it is denoted p_i. Likewise, the time associated with node i is denoted t_i. An arc (i, j) is included in B if and only if $t_j - t_i = t_{p_i,p_j}$, or if $i = s$ and $t_j = 0$, or if $j = f$ and $t_i = t_{\text{max}}$. This includes all possible eVTOL flight connections, as well as links the fictitious origin and destination nodes with each of the ports at the beginning and end of the time horizon, respectively. We allow for an eVTOL to remain stationary at port $i \in N$ for a single time step by defining $t_{ii} = 1$ for all $i \in N$. Note, H is an acyclic network. Denote the change in an eVTOL's charge level associated with arc $(i, j) \in B$ with $e_{ij} \in \mathbb{R}$, which may correspond to an increase or a decrease. Let $\delta^+(i) \subset M \cup \{s, f\}$ and $\delta^-(i) \subset M \cup \{s, f\}$ denote the set of outgoing and incoming nodes for node $i \in M \cup \{s, f\}$, respectively. That is, $\delta^-(i)$ denotes those j such that $(j, i) \in B$, and $\delta^+(i)$ denotes those j such that $(i, j) \in B$. Let $\mathcal{F}(c) \subseteq B$ denote the set of possible flights for customer c. This will be the collection of arcs $(i, j) \in B$, such that $p_i = o_c$, $p_j = d_c$, and $t_i \in T_c$. Denote the set of all possible flights $\mathcal{F} = \bigcup_{c \in C} \mathcal{F}(c)$.

4.4.2 Three-Index Formulation

For each arc $(i, j) \in B$, and for each eVTOL $k \in K$, let x_{ij}^k be a binary variable equal to 1 if and only if eVTOL k traverses arc (i, j). For each customer $c \in C$, and for each $(i, j) \in \mathcal{F}(c)$, let the binary variable y_{ij}^c be equal to 1 if and only if customer c flies along arc (i, j). For each $i \in M \cup \{s, f\}$, and for each eVTOL $k \in K$, let $q_i^k \in \mathbb{R}$ indicate the charge level of eVTOL k upon arrival at node i. If eVTOL k does not visit node i, the quantity is meaningless. With these variables, we can formulate the UAMP with the following mixed-integer program, which we denote MIP3 to reflect the three indices on the decision variable x_{ij}^k:

$$\max \sum_{c \in C} \sum_{\{(i,j) \in \mathcal{F}(c)\}} y_{ij}^c \tag{4.1}$$

$$\text{s.t.} \quad \sum_{j\in\delta^+(i)} x_{ij}^k - \sum_{j\in\delta^-(i)} x_{ji}^k = \begin{cases} 1 & i = s \\ 0 & i \in M\backslash\{s,f\} \\ -1 & i = f \end{cases} \quad \forall i \in M \cup \{s,f\}, k \in K \tag{4.2}$$

$$\sum_{\{c|(i,j)\in\mathcal{F}(c)\}} y_{ij}^c \leq \sum_k S x_{ij}^k \quad \forall (i,j) \in \mathcal{F} \tag{4.3}$$

$$\sum_{(i,j)\in\mathcal{F}(c)} y_{ij}^c \leq 1 \quad \forall c \in C \tag{4.4}$$

$$q_0^k = Q_0^k \quad \forall k \in K \tag{4.5}$$

$$q_j^k \leq (q_i^k + e_{ij}) + M(1 - x_{ij}^k) \quad \forall (i,j) \in H, k \in K \tag{4.6}$$

$$Q_- \leq q_i^k \leq Q_+ \quad \forall i \in M, k \in K \tag{4.7}$$

$$x_{ij}^k \in \{0,1\} \quad \forall (i,j) \in H, k \in K \tag{4.8}$$

$$y_{ij}^r \in \{0,1\} \quad \forall (i,j) \in \mathcal{F}(c), c \in R \tag{4.9}$$

$$q_i^k \in \mathbb{R} \quad \forall i \in M, k \in K \tag{4.10}$$

The objective function 4.1 maximizes the number of customers that are served. Constraints 4.2 ensure that each vehicle k follows a path through the time-expanded network H, beginning at s and ending at f. Constraints 4.3 ensure that for each possible flight arc, the number of passengers traveling along that arc is bounded from above by the number of eVTOLs taking the same arc, multiplied by the uniform eVTOL capacity. Constraints 4.4 ensure that each customer is served at most once. Constraints 4.5 initialize the charge level of each eVTOL. Constraints 4.6 ensure that the charge level of an eVTOL evolves according to its path. This is achieved with a 'Big M' term, where, e.g., $M = Q_+ + \max_{(i,j)\in B} e_{ij}$. Constraints 4.7 bound the charge levels. Constraints 4.8 and 4.9 ensure binary-valued eVTOL flow and passenger flow variables, respectively, and constraints 4.10 ensure real-valued charge levels. There are a number of extensions that can be incorporated into this formulation, including customer groups, soft time windows, and battery swaps. We describe these modifications in Appendix B.1.0.3,B.1.0.2, and B.1.0.4, respectively.

4.4.3 Two-index Heuristic

The charge constraints significantly complicate the problem, and, therefore, severely limit the problem sizes that can be solved using MIP3 within a reasonable computation time. One idea is to relax the charge constraints, solve the problem, and, then, repair the solution to make it charge-feasible. When the charge constraints are relaxed it suffices to aggregate the arc variables to obtain a two-index formulation, i.e., aggregate the eVTOL flow variables from MIP3, $x_{ij} = \sum_{k \in K} x_{ij}^k$. The integer-valued variable x_{ij} in a two-index formulation thus corresponds to the number of eVTOLs traveling along arc (i, j). The resulting integer program (that we refer to as MIP2) is described in Appendix B.2. The solution to MIP2 provides a set of K throughput-maximizing paths through the network, which may not satisfy charge constraints. To obtain a charge-feasible solution, we use the solution to MIP2 to sparsify the network and then apply MIP3.

The process is as follows. We first solve MIP2 and obtain an optimal solution $x^* = \{x_{ij}^*\}_{(i,j) \in B}$. We then construct a sparsified network B^0. An arc $(i, j) \in B$ is included in B^0 if and only if $x_{ij}^* > 0$ or $p_i = p_j$ and $t_j = t_i + 1$. That is, we only keep the arcs from the solution of MIP2 as well as the arcs corresponding to remaining at a port for consecutive time steps. Keeping these latter arcs ensures charge feasibility. MIP3 can be run on this dramatically sparsified network. We refer to the process of sparsifying in this fashion, and then running MIP3 on the sparsified network as MIP2⇒3.

Although this approach dramatically improves upon MIP3 in terms of the size of the problems solved, it's efficacy is limited as the scale of problems grows even larger (a detailed discussion of our computational experience with MIP2⇒3 is provided in Appendix B.2.0.1). Instead, to improve our ability to find high-quality solutions to larger instances, we pursue a path-based approach in the next section.

4.5 Path-Based Approach

We now describe a path-based approach to solve the UAMP. Path-based approaches generally lend themselves to efficient column generation schemes, as well as lead to stronger LP-relaxations than arc-based counterparts. We present the path-based formulation in Section 4.5.1 and develop the corresponding column generation problem in Section 4.5.2. We then present the subproblem algorithm in Section 4.5.3, the manner in which initial paths can be determined in Section 4.5.4, and the branching rules we employ in the event of fractional solutions in Section 4.5.5. Finally, Section 4.5.6 describes a network sparsification heuristic that we apply in the path-based approach to speed up the procedure.

4.5.1 Path-Based Formulation

Let C continue to be the set of customers, and V the fleet of eVTOLs. Let R be the set of all charge-feasible eVTOL paths. That is, the set of all paths through the network H from s to f such that all charge constraints are respected. Each route r is associated with a set of customers it serves. For each route, we define the binary-valued vector $s_r = \{s_r^c\}_{c \in C}$, where $s_r^c \in \{0, 1\}$ is 1 if and only if customer c is served by route r. We refer to s_r as the service vector of route r. For each path $r \in R$, let $x_r \in \{0,1\}$ be 1 if and only if route r is selected. Let $y_c \in \{0,1\}$ be 1 if and only if customer c is served. We may reformulate the UAMP as the following integer program:

$$\max \sum_{c \in C} y_c \tag{4.11}$$

$$\text{s.t. } y_c - \sum_{r \in R} s_r^c x_r \leq 0 \qquad \forall c \in C \tag{4.12}$$

$$\sum_{r \in R} x_r = |V| \tag{4.13}$$

$$x_r \in \{0,1\} \qquad \forall r \in R \tag{4.14}$$

$$y_c \in \{0,1\} \qquad \forall c \in C \tag{4.15}$$

We denote this integer program MP, for master problem. The objective 4.11 maximizes the number of customers served. Constraints 4.12 ensure customers are only counted as served if a route which serves them is selected. Constraint 4.13 ensures the number of eVTOLs is equal to $|V|$. Constraints 4.14 and 4.15 ensure binary-valued route selection and customer service variables, respectively. The above formulation may be relaxed as a linear program by replacing constraints 4.14 and 4.15 with the constraints below:

$$y_c \leq 1 \qquad \forall c \in C \tag{4.16}$$

$$y_c \geq 0 \qquad \forall c \in C \tag{4.17}$$

$$x_r \geq 0 \qquad \forall r \in R \tag{4.18}$$

Constraints 4.16 prohibit any customer from being counted more than once, and constraints 4.17 and 4.18 ensure nonnegativity. We need not bound the x_r variables by 1, as there is no benefit obtained with solutions that have $x_r > 1$ (i.e., if $x_r > 1$ occurs, we replace x_r by 1, remaining feasible and use the excess above 1 to bring in another route variable(s)). However, as a linear relaxation, fractional solutions are a possibility. We denote this linear relaxation L-MP, for linear master problem.

4.5.2 Column Generation

The main challenge in the path-based approach is the size of R, the set of all charge-feasible paths. This set is generally very large, and, therefore, the MP formulation consists of too many variables to be efficiently solved using a simplex/branch-and-bound based solver. Instead, we employ a column generation approach, by first selecting a small subset of paths, $R^0 \subset R$. We then solve the L-MP, restricted to this subset of paths. We denote this restricted problem L-RMP, for linear restricted master problem. We denote the corresponding problem with binary constraints the RMP. Solving the L-RMP results in a set of dual variables associated with each constraint. We can then use these variables to identify 'useful' paths. That is, those paths which, if included in the formulation, would lead to an increase in objective value. These paths are added to R^0, and the procedure is repeated. This can continue until no more useful paths can be identified, or some other stopping criterion is satisfied.

We now explicitly develop this idea. Let the dual variables α_i, β, and γ_i correspond to constraints 4.12, 4.13, and 4.16, respectively. Taking the dual of the path-based formulation, we have:

$$
\begin{aligned}
\min\ & |V|\beta + \sum_{c\in C}\gamma_c \\
\text{s.t.}\quad & \alpha_i + \gamma_i \ge 1 && \forall c\in C \\
& \beta - \sum_{c\in C} s_r^c\alpha_c \ge 0 && \forall r\in R \\
& \alpha_c \ge 0 && \forall i\in C \\
& \gamma_c \ge 0 && \forall i\in C.
\end{aligned}
\tag{4.19}
$$

If constraint 4.19 is violated for any route $r \in R$, it indicates that route has a positive reduced cost, implying the primal solution must be suboptimal. Thus, we seek routes r such that:

$$\beta - \sum_{c\in C} s_r^c\alpha_c < 0$$

and the subproblem can be stated as:

$$\max_{r\in R}\left\{\sum_{c\in C} s_r^c\alpha_c\right\}.$$

Thus, we seek the route $r \in R$ such that the inner product of its service vector, s_r, and $\alpha = \{\alpha_c\}_{c\in C}$ is maximized. This can be stated as an elementary shortest path problem with resource constraints, a well-studied problem [see 92]. This is done in the following way. For each customer $c \in C$, each arc $(i,j) \in \mathcal{F}(c)$ is given a weight of $-\alpha_c$. If an arc (i,j) corresponds to a flight arc for multiple customers (i.e., $(i,j) \in \mathcal{F}(c_1)\cap\mathcal{F}(c_2)\cap\ldots\cap\mathcal{F}(c_k)$), the weight is simply the negative of the sum of the corresponding α values (i.e., $-(\alpha_{c_1}+\ldots+\alpha_{c_k})$). If the number, k, of such customers is greater than the uniform eVTOL capacity (i.e., $k > S$), then the arc weight is the negative of the sum of the S greatest corresponding α values. All other arcs in the network are given a weight of 0. We then seek a path from s to t of minimum weight, where the path weight is the sum of the arc weights along the path. Of course, multiple paths can have the same weight, and indeed, many paths can share a service vector despite taking a different route through H. Because our underlying network is acyclic, all feasible paths are acyclic, and, thus, we may instead solve the shortest path problem with resource constraints (SPPRC), a significantly easier problem (the elementary shortest path problem with resource constraints requires that the path be loopless, while the shortest path with resource constraints has no such restriction). We implement a label-setting algorithm to solve this problem, which we describe in Section 4.5.3.

4.5.3 Subproblem

We seek to identify paths of positive reduced cost to be added to the reduced master problem, which amounts to solving the shortest path problem with resource constraints on the time expanded network $H = (M, B)$. A standard way to solve shortest path problems is to use a label-setting algorithm, which we implement here. For an introduction to the method, see Ahuja et al. [62]. The main data structure in the algorithm is the label, which corresponds to the state of an eVTOL at a particular point on its path, having completed some sub-path through the network. Labels are identified by the current node, current charge level, current score, and a pointer to the previous label. We establish two lists of labels: the unprocessed labels, U, and the processed labels, P. We initialize by adding the starting label, l_0, to U, where $l_0 = [s, Q_0, 0, \emptyset]$. That is, the starting label is assigned the current node s, the initial charge, a score of zero, and no previous label. While U is nonempty, we choose a label $l \in U$. Let i_l, q_l, and z_l be the node, charge level, and score associated with label l, respectively. We then 'treat' this label. This is done by observing all $j \in \delta^+(i_l)$ (the outgoing nodes of i_l), and creating the new label, $l' = [j, \min(Q_+, q_l + e_{ij}), z_l + z_{ij}, l]$, where z_{ij} is the the weight of the arc (the sum of the reduced costs of the customers that can be served along the arc, or the greatest such sum if the number of customers exceeds the capacity). If $q_l + e_{ij} < Q_-$, the label is thrown away. Otherwise, the label is added to U. Once l is treated, it is added to P. This continues until U is empty. Once U is empty, P is searched for the label $l \in P$ corresponding to a complete path that maximizes z_l. That is, we identify $\text{argmax}_{z_l}\{l \in P \mid i_l = f\}$. This path will be the solution to the SPPRC. We can identify the path corresponding to this label by backtracking through the pointers to previous labels. We can likewise identify the customers served along the path by, for each arc, identifying those customers who can be served along the arc. If the number of such customers exceeds the capacity, we choose the S customers of greatest reduced cost.

There are a couple of considerations that can speed up this algorithm. We can employ a dominance filter to the labels added to U. A new label, l^*, is added to U only if it is not dominated by any label $l \in U \cup P$. A label a dominates another label b if and only if $i_a = i_b$ and $q_a \geq$ and $z_a \geq z_b$. A useful feature of the label-setting algorithm is that many paths of positive reduced cost can be produced in one pass. Rather than only extracting the path of maximum reduced cost, we can instead extract several paths of positive reduced cost at a time.

4.5.4 Initialization

In the column generation approach, a set of starting paths must be identified. It has been noted [by, for example, 61] that the choice of initial paths is important, as they influence the dual variables identified by L-RMP, and, thus, the new paths produced. Efforts must be made to produce ‘good’ paths.

We implement a simple greedy initialization procedure. First, we randomly select a starting port and initialize the charge level (typically setting the charge to full). Then, we consider each possible choice of a next node, determined by the arcs in the network, as well as the current charge level. We choose the arc of the greatest score. The score is determined the same way as in the label-setting method, where each customer’s α value is set to 1. We then add the arc to the path and move to the corresponding node. In the case of a tie in score, we randomly choose between the tied arcs. We continue until reaching node f. This greedy heuristic can also be used as a heuristic to the subproblem, as an alternative to the labeling algorithm discussed in Section 4.5.3. That is, we can apply the greedy heuristic to find ‘good’ paths with positive reduced cost, and resort to the exact labeling procedure if the greedy algorithm yields no paths.

4.5.5 Branching

We branch on the fractional x_r variables, by creating two nodes, one corresponding to $x_r = 0$, the other $x_r = 1$. This corresponds to partitioning solutions into those that prohibit route r (as well as all routes that serve the same customers as route r) and those that include route r. In the latter case, the (pricing) subproblem becomes simpler, as the number of eVTOLs can be reduced by 1, and the customers served by route r may be removed from consideration.

In the former case, when x_r is enforced to be 0, we must prohibit the corresponding variable from entering the solution. Note that other routes may serve the same set of customers as route r. This can happen, for example, by shifting the departure times. We say two routes are equivalent if they serve the same set of customers, i.e., r' is equivalent to r'' iff $s_{r'} = s_{r''}$. We remove all routes equivalent to r from the L-RMP. Furthermore, to prevent the entry of an equivalent route into the L-RMP, we add the set of customers served by r to a list of prohibited customer sets. Then, for each route in P (the set of complete routes found by the labeling algorithm), we check if the set of customers served by the route is in the list. If so, the route is removed from consideration.

4.5.6 Network Sparsification

To improve the run times of the above approaches, we implemented a network sparsification procedure to further limit the size of the solution space. For a customer c, the arcs $\mathcal{F}(c)$ are her **passenger flight arcs**. Let $t_c^a = \min\{t|t \in T_c\}$, $t_c^b = \max\{t|t \in T_c\}$ be the first and last time values at which customer c can depart from her origin port, respectively. Let the two arcs $\hat{\mathcal{F}}(c) = \{(i,j) \in \mathcal{F}(c) \mid t_i \in \{t_c^a, t_c^b\}\}$ be the **extreme flight arcs**. For a customer c, the arcs $\mathcal{A}(c) = \{(i,j) \in B \mid p_j = o_c, t_j \in T_c, p_i \neq p_j\}$, are the customer's **anticipatory arcs**. These are those arcs that are traveling from another port to the customer's origin within the customer's time window. Let the arcs $\hat{\mathcal{A}}(c) = \{(i,j) \in \mathcal{A}(c) \mid t_j \in \{t_c^a, t_c^b\}\}$ be the customer's **extreme anticipatory arcs**. Denote the arcs $\{(i,j) \in B \mid p_i = p_j, t_i + 1 = t_j\}$ as the **stay put arcs**. We limited the time expanded networks in our data sets to only include the extreme passenger flight arcs, extreme anticipatory arcs, stay put arcs, and the arcs connecting the fictitious starting and ending nodes to the network. Figure 4.1 illustrates this sparsification for a 3-customer, 3 port, 20 time step problem. Left: The original network, which includes every possible eVTOL flight. The passenger flight arcs are in bold. One customer wants to go from Port 1 to Port 3, departing at t = 2, 3, or 4. Another wants to go from Port 2 to Port 1, departing at either t = 6 or 7. A third wants to travel from Port 2 to Port 1, departing at t = 10, 11, or 12. Right: The sparsified network, with only the extreme passenger flight arcs, extreme anticipatory flights, and stay put arcs. Extreme passenger flight arcs are in bold, and extreme anticipatory flights are dashed. In this example, the number of arcs is reduced from 167 to 80.

If there is no upper limit on the eVTOL charge level, and all flight/anticipatory arcs are included (rather than just the extreme arcs), we can prove the sparsification does not cut off the optimal solution (see Appendix B.3). We place an upper bound on the charge level, and only use the extreme flight/anticipatory arcs, and, thus, do not have this guarantee. However, experiments on synthetic data have shown there is little loss in objective value, but a considerable speed up in run time, when using this procedure. In particular, as the number of customers increases, the loss in objective value decreases.

4.6 Computational Results

We now describe our computational experience with the arc-based and path-based approaches. These include experiments on simulated data sets and a case study using Washington D.C. taxicab data. Since the instances are difficult to solve to optimality, a major goal in our experiments was to develop a heuristic variant of the path-based approach that is capable of finding high-quality solutions for the size of instances that may be expected when UAM service is rolled out. Our experiments were run on a Windows computer, using an Intel(R) Core(TM) i5-7400 CPU @

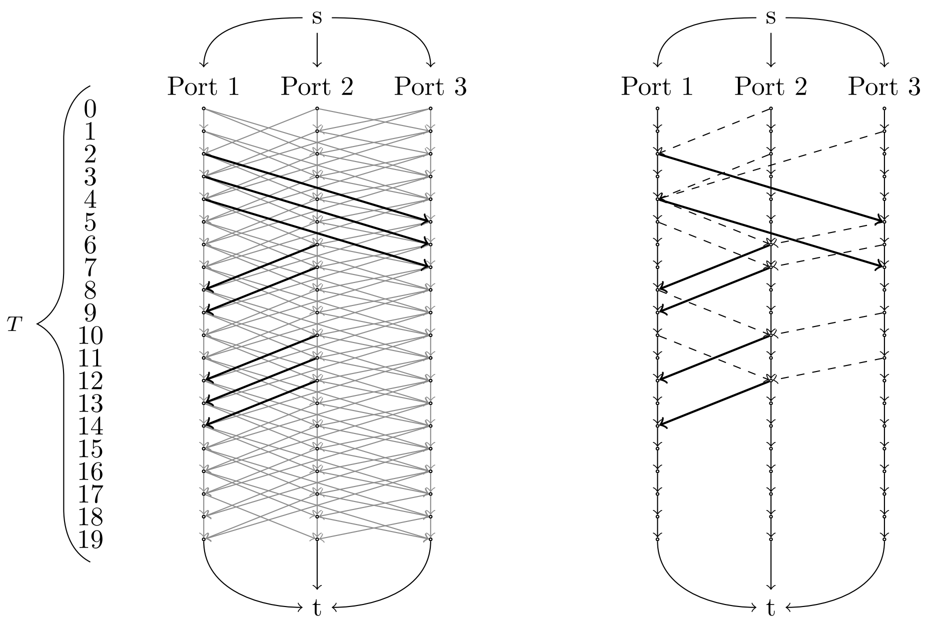


Figure 4.1: Sparsification procedure.

3.00GHz processor. Our codes were implemented in Python 3.7, and Gurobi 9.0 was used to solve all linear programs.

4.6.1 Simulated Data Sets

We developed problem sets to evaluate the relative performance of our solution methodologies. We first establish a discrete time horizon, $T = \{1, 2, \ldots, t_{\max}\}$, partitioned into minute-long intervals, so that one time step corresponds to one minute. We then initialize a $\frac{|T|}{10} \times \frac{|T|}{10}$ grid, in which $|P|$ ports are then randomly placed. The distance between ports i and j, d_{ij}, is Euclidean, rounded up to the nearest integer. The corresponding travel time, t_{ij}, between the ports is taken to be equal to d_{ij}. Each customer is randomly assigned an origin port and a (different) destination port, as well as the uniform time window length $W \in \mathbb{Z}$ minutes. That is, each is willing to wait for W minutes after the beginning of her time window. The beginning of each customer's time window then is randomly drawn from $T' = \{0, 1, \ldots, t_{\max} - W - t^M\}$, where

$t^M = \max_{i,j \in N} t_{ij} \in \mathbb{Z}^+$ is the longest of all the pairwise travel times. This ensures all flights arrive at their destination prior to the end of the time horizon. Charge transition values between nodes i and j, e_{ij}, are established as follows:

$$e_{ij} = \begin{cases} -\psi t_{p_i,p_j} & p_i \neq p_j \\ \phi & p_i = p_j \text{ and } t_j - t_i = 1 \end{cases}$$

where ψ and ϕ are the charge depletion and recharge levels per time step, respectively. Charge transitions are, thus, linear with respect to time. There are many parameters involved in the UAMP (number of ports, eVTOL speed, eVTOL capacity, number of eVTOLs, number of customers, passenger waiting time, etc.) that at this time vary based on eVTOL manufacturer (since most eVTOLs are still in the early concept phase with prototypes yet to be manufactured and certified), and market assumptions. Consequently, after reviewing industry white papers and studying several papers [see 93, 94, 95] describing specifications of the different eVTOLs in development, we chose a set of reasonable default values that the parameters could assume for testing purposes. Table 4.1 provides these values, along with their symbolic notation.

We created a set of smaller test instances, that we refer to as Problem Set A, that the branch-and-price approach was largely able to solve to optimality. This baseline family of problems has 4 ports and 4 eVTOLS, and the number of customers range from 5 to 200 in steps of 5 (a total of forty problems). We then introduce a larger (in terms of the port and eVTOL infrastructure) and more difficult family of problems, which considers 8 ports and 8 eVTOLs, with the number of customers ranging from 50 to 300 in steps of five (a total of fifty problems); we refer to these as Problem Set B. The branch-and-price approach was no longer viable for Problem Set B,

Parameter	Notation	Default Value
Time Horizon Length	$t_{\max}$	60 Time Steps
Time Step Size	τ	1 minute
Time Window Width	W	3 minutes
Maximum Charge Level	Q_+	100
Minimum Charge Level	Q_-	0
Initial Charge Level	Q_0	100
Charge Depletion Per Time Step	ψ	5
Recharge Per Time Step	ϕ	10
eVTOL Capacity	S	6

Table 4.1: Parameter values. These were the fixed values used in our experiments.

necessitating the use of column generation as a heuristic to obtain solutions for these instances. We have largely focused our evaluation of algorithms (in terms of how they scale) by varying the number of customers. This is because (i) the market demand is a big uncertainty and there is great value to UAM operators in understanding how solutions vary as the number of customers varies, and (ii) we found that the performance of our approaches was most sensitive to the number of customers (as compared to the number of eVTOLS or number of ports).

4.6.2 Results on Simulated Data Sets

We now present our findings with the arc-based and path-based procedures when applied exactly on Problem Set A. We then focus on the larger instances in Problem Set B, and experiment with heuristic versions of the path-based approach (for example, by limiting column generation to the root node). For ease of visualization, we present our results with figures. Tables including run times and objective values for each of the figures are included in Appendix B.5. All run times are in units of seconds.

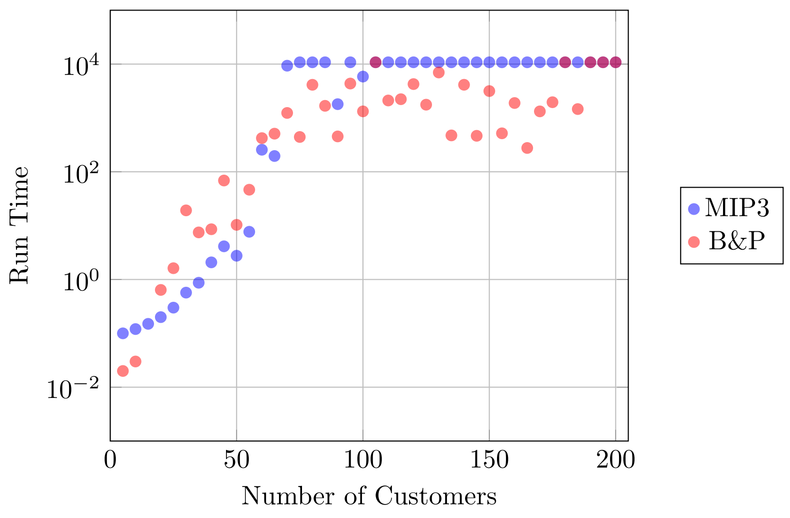


Figure 4.2: MIP3 and B&P run time on Problem Set A.

4.6.2.1 Comparing the Arc- and Path-Based Formulations as Exact Approaches

We compared the Three-Index formulation (MIP3) presented in Section 4.4.2 and the Branch-and-Price method (B&P) presented in Section 4.5 on Problem Set A, with a time limit of three hours. Figure 4.2 displays the results. While MIP3 appears more efficient for smaller instances, at around 70 customers, B&P outperforms it, and is able to produce and prove the optimal solution for all but five instances (when $|C| = 105, 180, 190, 195$ and 200). Both methods exhibit high variability in run time, with problems of similar size showing run times sometimes differing by an order of magnitude. This reveals a high level of sensitivity to the randomly determined customer data (i.e., origin-destination pairs and departure times).

4.6.2.2 Column Generation Heuristics

A popular heuristic version of path-based formulations is to limit column generation to the root node. Generation of columns at each node along the branch and bound tree can be a time consuming process, and it may be the case that such generation is unnecessary if, ultimately,

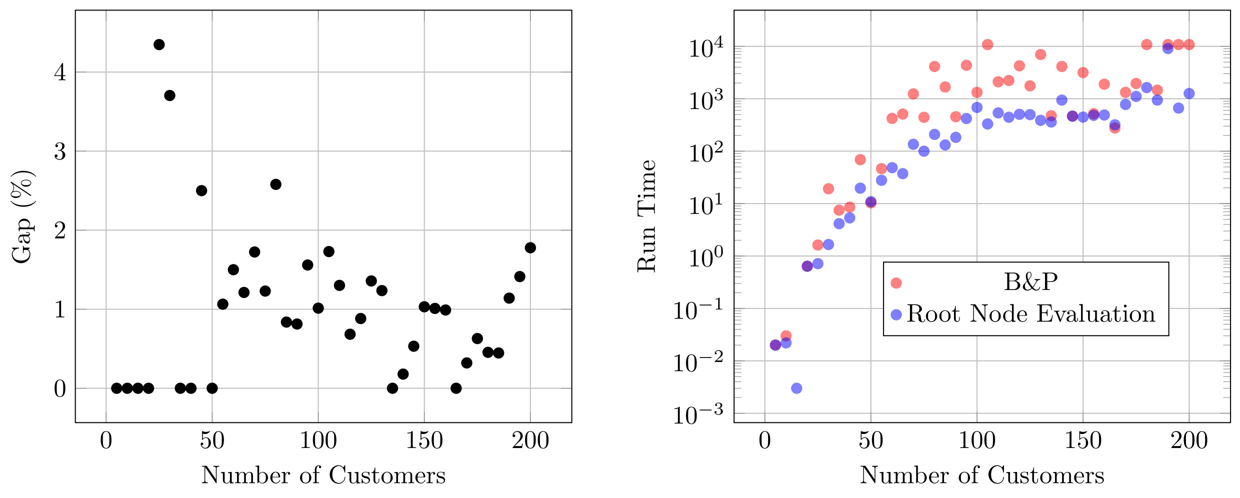


Figure 4.3: Optimality gap at the root node and run times for B&P on Problem Set A.

the optimal columns are already in the RMP. In Figure 4.3, we present justification for such a heuristic. On the left, we plot the optimality gap (between the upper and lower bounds) for B&P upon completing the root node on Problem Set A. We observe that, for the larger problems, the gap is less than 2%. On the right, with the y-axis shown in logarithmic scale, we show the run times for B&P and the run times to complete solving the RMP at the root node. We observe the bulk of computation time in B&P is spent closing a 2% gap, which can justify the heuristic of only generating columns at the root node.

We implemented several column generation based heuristics. First, we only generated columns at the root node. When no more columns of positive reduced cost can be identified, the program performs traditional branch and bound to find the optimal solution, subject to only having the columns generated at the root available. This heuristic is denoted RCG, for root node column generation. We then considered an extension of RCG, where column generation is not only constrained to the root node, but also stops generation after a certain number of columns have been produced. In Figure 4.4, we demonstrate how the marginal improvement of solutions diminishes as more columns are generated. Using ten instances of a 100 customer,

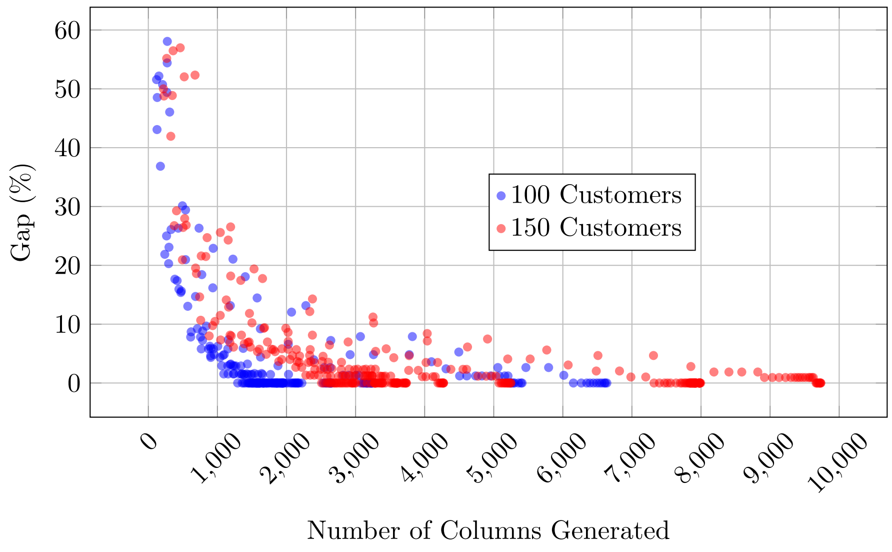


Figure 4.4: Optimality gap versus number of columns generated.

8 port, 8 eVTOL problem (these are distinct from Problem Set B), we record the gap from the best known solution (computed beforehand) as well as the number of columns generated so far. We do the same for ten instances of a 150 customer, 8 port, 8 eVTOL problem (again distinct from Problem Set B). We plot a point for each time the subproblem finishes, reporting the number of columns generated as well as the current gap from the optimal solution (computed beforehand using B&P). We observe that the 100 customer cases find their solutions after fewer columns generated than the 150 customer instances, with medians of 1782.5 and 3495 columns, respectively. Furthermore, we notice the ‘tailing off’ effect often seen in column generation approaches, with quick early improvement followed by slow incremental progress. We selected the limits of 1000, 3000, and 5000 columns, and denote the resulting heuristics RCG1000, RCG3000, and RCG5000.

Figure 4.5 presents the optimality gaps and run times for RCG, RCG1000, RCG3000, and RCG5000 on Problem Set A (we use Problem Set A in order to compare with the exact solutions obtained by B&P). We also include run times for B&P. The optimality gaps for the five problem

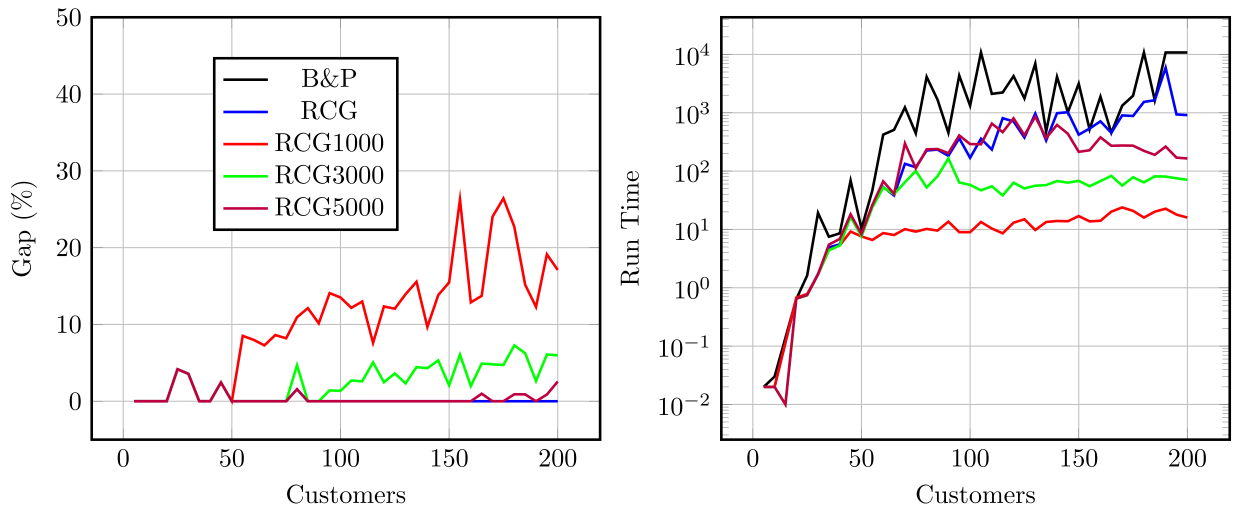


Figure 4.5: Performance of the column generation based methods on Problem Set A.

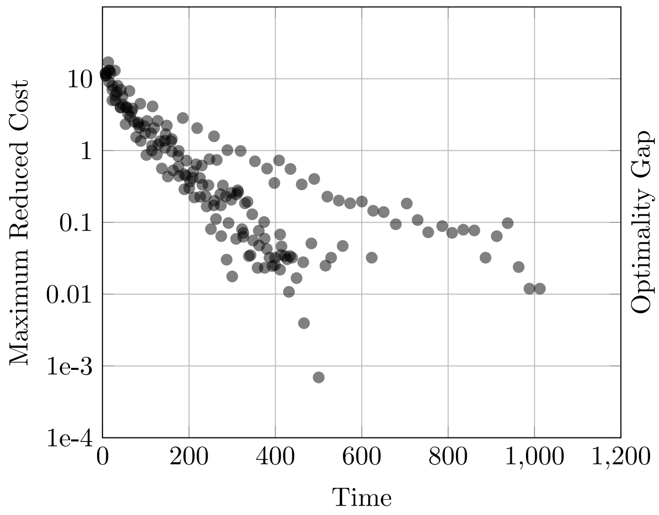
sizes for which B&P could not produce an optimal solution within three hours are calculated with respect to the best known solution. We observe the gap for RCG is very small (mostly zero) while requiring much less time than B&P. Meanwhile, RCG1000, while fast, produces a large optimality gap. As the number of columns produced increases, naturally, the gap goes down, and the run time goes up. Setting the number of columns to 5000 seems to find a good balance.

We observe the noted 'tailing off' effect of the column generation procedure, where the reduced costs of the generated columns diminish with the number generated, and the improvement in the upper and lower bounds slows. Consequently, we experimented with a strategy where we specify a threshold value such that if the maximum reduced cost of the latest batch of columns is below the threshold, we stop generating columns. Effectively, we determine at what point to treat very small reduced costs as 0. We applied this to both the B&P procedure (losing the optimality guarantee), and to RCG. To this end, we experimented with a set of five 150 customer, 8 port, 8 eVTOL problems. These instances are distinct from Problem Set B and the instances in Figure 4.4). We recorded the maximum reduced cost, as well as the optimality gap, upon each pass of the LRMP. Figure 4.6 displays plots of this investigation. The left plot shows the value of the maximum reduced cost of the latest batch of columns found by the subproblem versus the elapsed time. We observe the maximum reduced costs show roughly exponential decay. On the right is a plot of the current optimality gap versus the maximum reduced cost among the latest batch of columns.

Together, these plots reveal that, initially, useful columns are produced, with maximum reduced costs on the order of 10^{-2} to 10^{1}. As time passes, the magnitude of the reduced costs decreases. Our investigation indicates that columns of reduced cost less than 0.01 do not improve the objective value.

We also noticed that reduced costs can plummet to the order of 10^{-14}, suggesting error due to floating point arithmetic. Two of the five instances spend a significant portion of the computation time identifying these columns of such extremely small reduced cost before terminating (these instances account for the cluster of points on the bottom left of the right plot). We also observed that once these columns are being produced, the objective value is already at the maximum value it will reach. Thus, we treated reduced costs below 10^{-6}, as being the same as 0. We apply this modification for all procedures involving column generation (including the instances solved using B&P previously).

As a heuristic, we then considered threshold values of 10^{-1} and 10^{0} as a stopping criterion for RCG (rather than setting a limit on the number of columns to generate). We label the heuristics RCG-TH-0.1 and RCG-TH-1, respectively. Figure 4.7 compares RCG, RCG-TH-0.1 and RCG-TH-1 on Problem Set A. Both RCG-TH-0.1 and RCG-TH-1 show a small optimality gap for all problem sizes. The run times of RCG-TH-0.1 often improve on those of RCG by an order of magnitude, while the run times for RCG-TH-1 are, naturally, even better. The small optimality



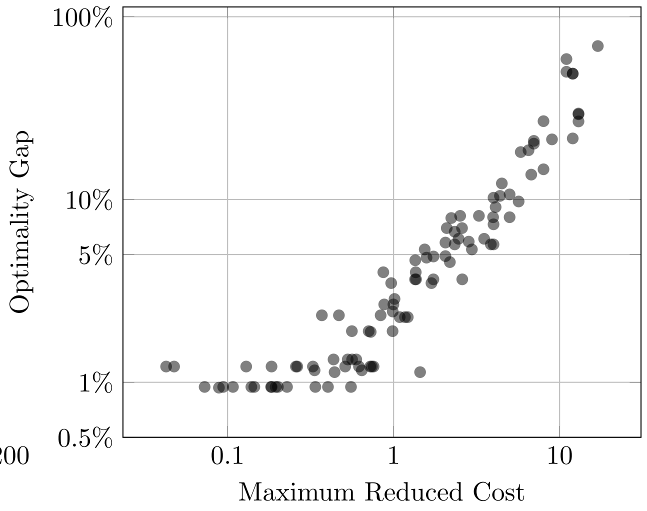


Figure 4.6: Comparison of maximum reduced costs with run times and optimality gaps.

gap and the significant time savings indicate RCG-TH-1 finds a good balance.

We then investigated the performance of the 'winners' from the previous experimentation (RCG, RCG5000, RCG-TH-0.1, and RCG-TH-1) on the larger data set, Problem Set B. The plot in Figure 4.8 reveals that RCG5000 eventually suffers in performance on larger instances. Meanwhile, RCG-TH-0.1 and RCG-TH-1 both provide high-quality solutions remaining close in objective value to the best solution found by RCG across all problem sizes, while yielding significant time savings. In particular, RCG-TH-1 appears to reduce the run time from RCG by an order of magnitude, while remaining within 2% of the best known objective value across all problem sizes.

4.6.2.3 Sensitivity to Time Step Size

Our model is built on a time-expanded network, where each port is paired with each point in time to create a 'port-time' node. This can be interpreted as modeling the operational practice of eVTOLs leaving only at discrete, regularly spaced times, say every five minutes. Clearly,

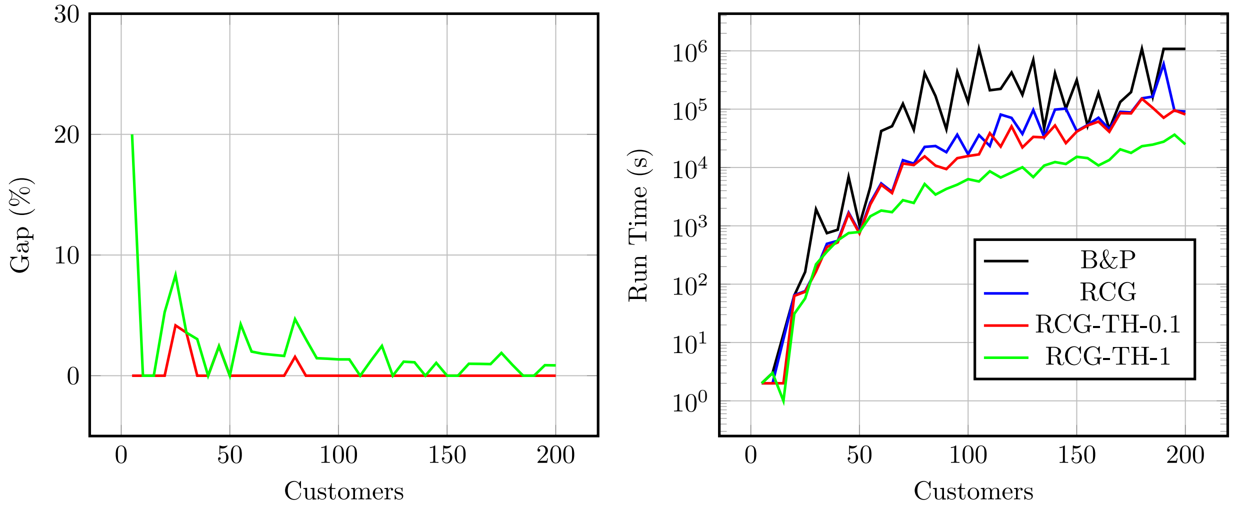


Figure 4.7: Performance of the threshold-based heuristics on Problem Set A.

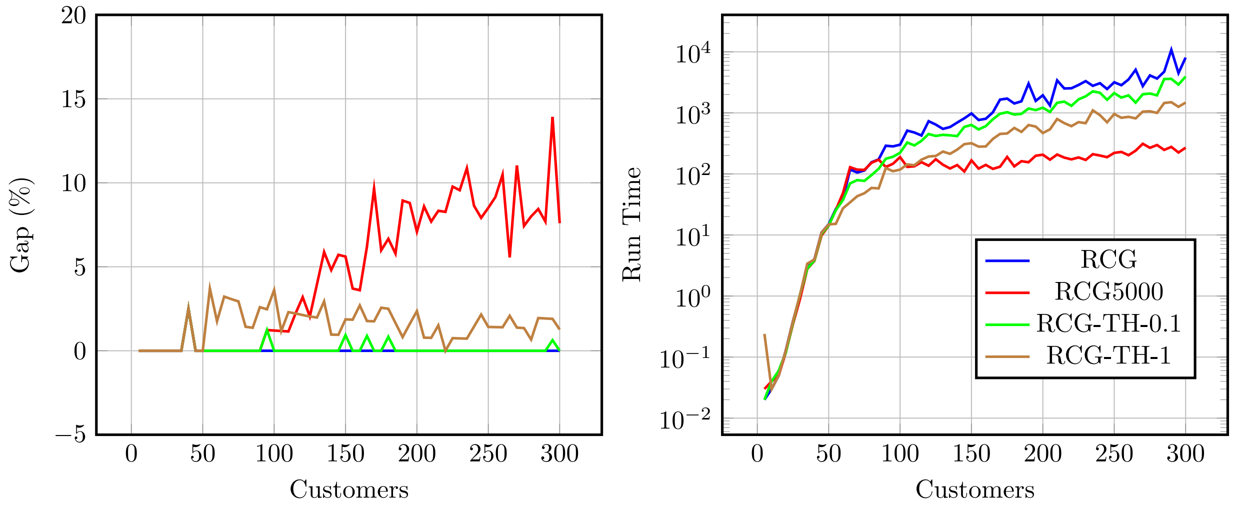


Figure 4.8: Performance of the better-performing methods on Problem Set B.

the finer the discretization of the time horizon (i.e., the shorter the spacing), the larger the resulting network, and, thus, the greater the computational demands. However, the increased flexibility of the denser network can theoretically lead to an improved objective value. In our initial experiments, we chose a time step size of one minute. However, in practice, this level of precision may be too ambitious. It may be unreasonable to expect all parties (eVTOLs and customers) to conform to any schedule right down to the minute, and, thus, in practice, a coarser discretization of time would serve as a buffer for small delays. The computational advantage is clear by observing that merely doubling the time step reduces the number of nodes by half, and similarly at least halves the number of edges in the network.

However, an issue arises when applying customer time window constraints. Because eVTOLs only depart at time steps, if a customer's time window is entirely contained within the width of a time step, the customer is unable to be serviced. For example, if the time horizon is partitioned into five minute intervals, such that possible departure times are 8:00 am, 8:05 am, 8:10 am, etc., a customer whose time window spans 8:06 am to 8:09 am would be impossible to serve. The best way to address this situation depends on the modeling approach.

One approach is to allow the customer to take the first departure after her window ends. For example, we allow the customer in the above example to take the 8:10 am flight. To be consistent, we must allow *all* customers to take the first departure after their windows end. For example, a customer whose window is 8:04 am to 8:09 am will also be allowed to take the 8:05 am or the 8:10 am flights. This artificial extension of time windows is straightforward to implement and seems justifiable from a practical perspective.

We note this approach makes it difficult to compare various discretizations with respect to the objective value. A consequence of allowing for the first departure after a window is that coarser discretizations have longer effective time windows, making the problem easier. Meanwhile, the reduced flexibility limits the space of solutions. The degree to which these effects cancel out one another is difficult to predict.

We conduct tests on the effects of increasing the time step size using this approach. We considered a set of ten 100 and ten 150 customer instances with 8 ports and 8 eVTOLs (these are distinct from the previously discussed instances). For these instances, we varied the time step size from 1 to 6 minutes, and solved each of the resulting problem instances using RCG. We measured the impact of the network losing flexibility by measuring the percent throughput compared to the one-minute time step case. Figure 4.9 plots the average percentages and average run times (averages are for the ten instances of a given time step size). We observe a linear decrease in objective percentage as the step size is increased. For example, when the time step is increased to two minutes, the objective value reached is, on average, around 95% of the value reached when the time step is one minute, and that value drops to 90% when the step is increased to three minutes. Meanwhile, the run times appear to decrease exponentially, dropping from 200 seconds to 70 seconds when going from one-minute to two-minute time steps. The plots indicate that, if one is willing to tolerate a certain decrease in objective value, the run time can be reduced significantly.

Increasing the time step substantially reduces computational time, thereby allowing one to address larger problems. To compare the scalability of the different approaches and assess their solution quality we created an augmented Problem Set B*. This data set contains the test instances from Problem Set B with 50, 100, 150, 200, 250, and 300, customers as well as larger 8 port, 8 eVTOL instances starting at 350 customers, going up to 900 customers in increments of 50 customers. Since RCG-TH-1 has a significant advantage over RCG-TH-0.1 in terms of

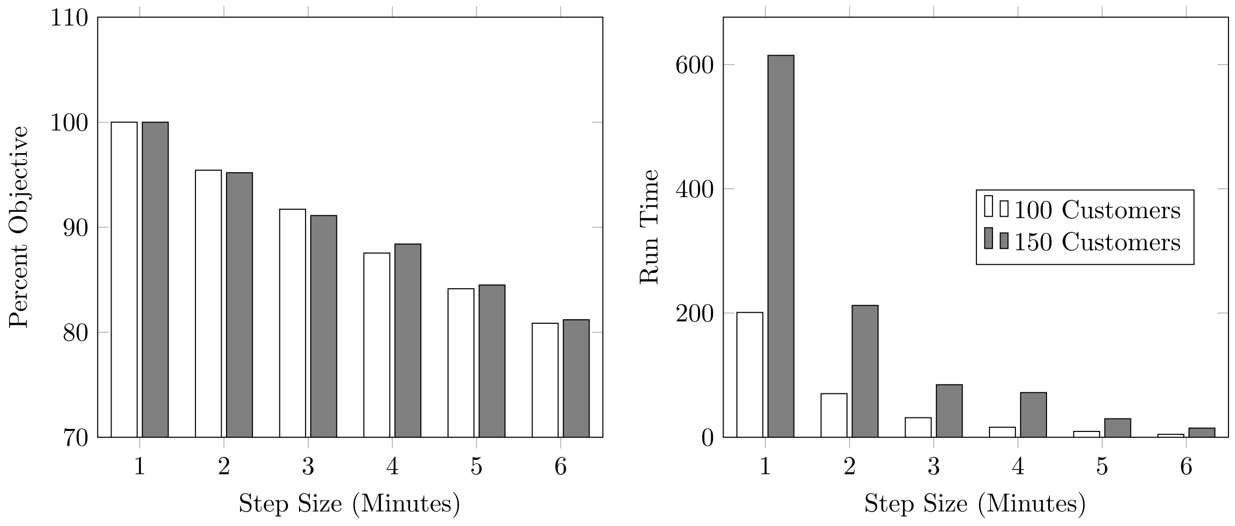


Figure 4.9: Sensitivity to time step size.

running time without a significant deterioration in objective value, for these large-scale instances in Problem Set B*, we dropped RCG-TH-0.1 from consideration. Instead, we compared RCG, RCG5000, RCG-TH-1, and a new variant of RCG-1hr on Problem Set B*. RCG-1hr limits root node column generation to one hour, and then branches, subsequently. Further, we apply the following heuristic procedure which significantly speeds up the column generation procedures.

After each run of the linear master problem, the $|V|$ largest x_r variables are identified. Each of these $|V|$ x_r variables can be set to 1, providing a feasible solution and, thus, a lower bound. We use the best lower bound identified in this way as the starting solution for each run of the linear master problem. We found this provides a significant speed up, especially as the number of columns gets large.

Figure 4.10 displays the results for RCG, RCG5000, RCG-TH-1, and RCG-1hr on Problem Set B*, using a time step size of two minutes. Notice that we are able to consider larger numbers of customers with the coarsened time step (indicating it could be a productive strategy to scale the approach). For instance, with the one-minute time step, RCG takes over two hours to solve

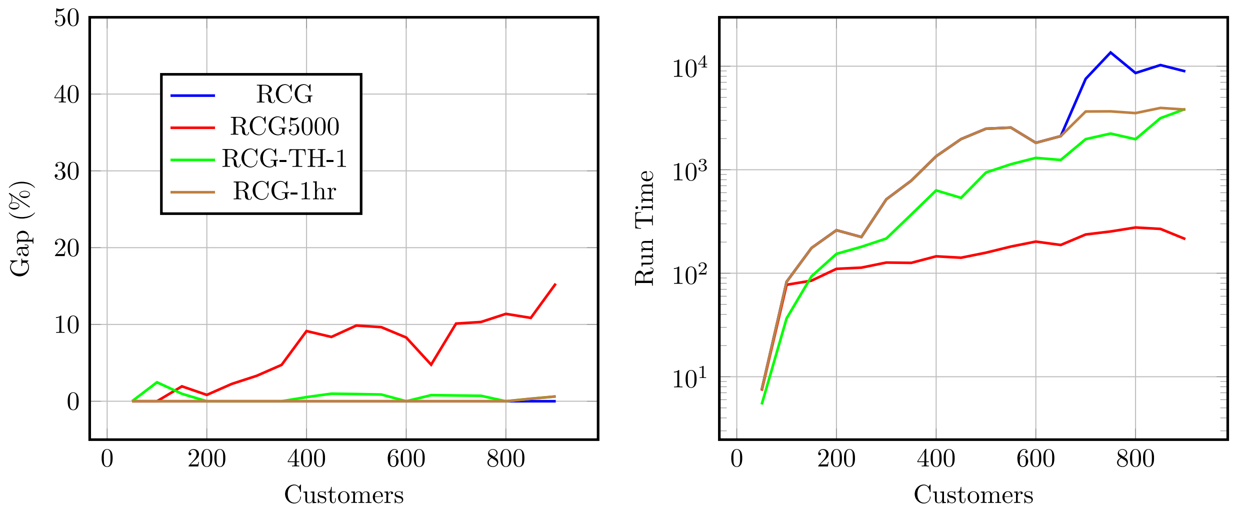


Figure 4.10: Performance of column generation methods on Problem Set B*.

the 300 customer instance, whereas, with the two-minute time step, it takes around 9 minutes. RCG and RCG-1hr are identical in behavior except for larger instances where RCG generates columns for more than an hour. Since the performance or RCG-1hr dominates RCG-TH-1 with respect to objective value and its run time is limited to (at most) a little over an hour by design, we recommend RCG-1hr as the column generation variant for large scale instances.

4.6.3 Case Study: DC Metro Area

We now transition to exploring the insights possible when applying our algorithms to a real-world problem. In this section, we model the UAMP on the DC metropolitan area and evaluate how various operational parameters affect solution quality.

4.6.3.1 DC Problem Set

We introduce a problem set which attempts to model the anticipated real-world setting. The District of Columbia Department of For-Hire Vehicles (DFHV) provides publicly-available taxi ridership records, including origin location, destination location, and trip duration for all taxi trips in a given year. For privacy reasons, departure times are provided only to the nearest hour. We pulled 5000 trips of duration greater than 30 minutes from this data set from the year 2017. This is because longer trips of this kind will be the most likely to be replaced by eVTOL trips. Then, to place the eVTOL ports, a k-means clustering algorithm [see 96] is run on the resulting set of nodes (that is, all origin and destination nodes). The resulting centers are the port locations. As origins and destinations are given in geographic coordinates, distances between nodes are given by the Haversine great-circle distance formula. A visualization of the taxi data, as well as the resulting port locations, is presented in Figure 4.11. In the plot, we represent a trip as a thin arc from its origin to its destination. Overlaid on the plot is the resulting placement of eVTOL ports as determined by the k-means clustering algorithm. We see from this plot that airport trips account for a significant portion of long taxi trips, and accordingly, each of the DC area's major airports has an eVTOL port nearby.The travel time between port i and j is calculated with $t_{ij} = \lceil d_{ij}/v \rceil + K$, where v is the eVTOL flight speed, and K is a fixed amount of time allotted for takeoff/landing/etc.

To construct a realistic test instance, we create a problem with 8 ports, a fleet of 20 eVTOLs, a set of 1000 customers, over a three hour horizon partitioned into 5 minute intervals Customer origins and destinations are sampled from the 5000 trips from the taxi data set, mapped to their nearest port.

Flight speeds are set at 240 kilometers per hour and time for takeoff/landing set to five minutes. Customer windows are set to be ten minutes long (which corresponds to two time steps, given a step size of five minutes). Requested departure times are spread uniformly over the time horizon (though restricted so that all possible flights arrive at their destination prior to the end of

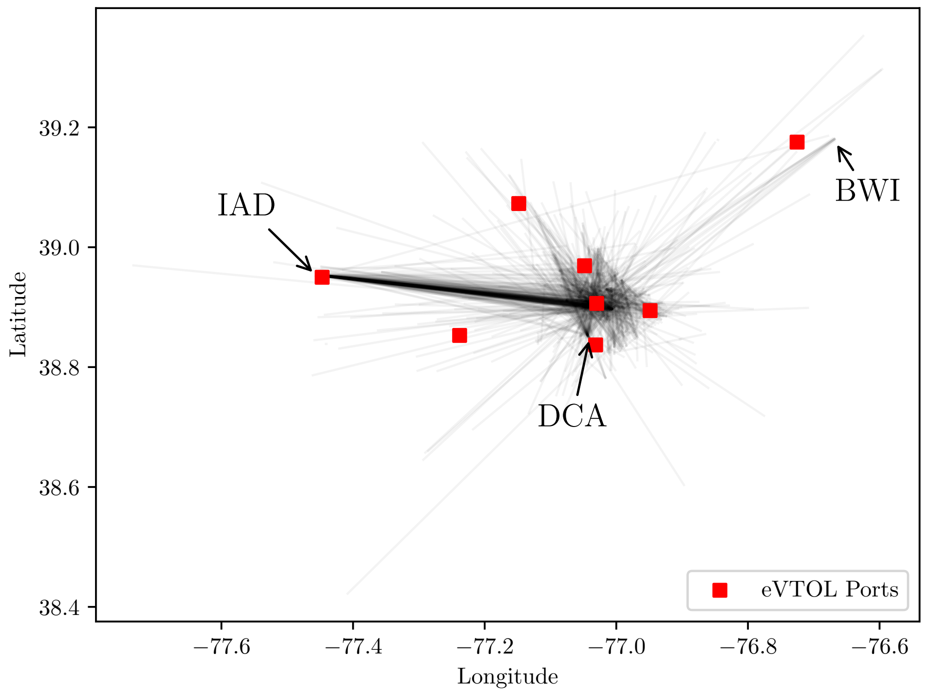


Figure 4.11: Visualization of the taxi ridership data.

the time horizon). The eVTOL charge depletion rates were set to 5% per five minutes of flight, and recharge rates to 10% per five minutes of charging. At the beginning of the time horizon, the eVTOLs are at full charge. [Recall we used industry white papers and 93, 94, 95, to help us determine specifications for eVTOLs.] We ran RCG-1hr on five instances of the above problem.

4.6.3.2 Results

We now describe several experiments that we conducted on the case study problem to assess the quality of the solutions generated by our path-based approach, and to assess some tradeoffs that managers are likely to be interested in. First, we were interested in the benefit in applying the path-based approach in contrast to a (simpler) greedy strategy. Table 4.2 presents the objective value found by RCG, with a one hour limit on column generation and a thirty minute limit on branching, compared with the objective values obtained when applying the greedy route generating procedure described in Section 4.5.4. We see that our path-based approach leads to a 30-50% improvement in the solution quality for problems of this size.

Battery management is an important operational restriction in eVTOL logistics. Operators of eVTOLs have an option (though it can be expensive) to swap batteries as needed, instead of waiting for the battery to be charged (if the next trip cannot be performed with the current charge). We wanted to see how much customer throughput can be increased when eVTOL operations are freed from battery recharging constraints. In effect, we are assuming there will always be a replacement battery available to swap at a port when necessary (we note that swapping is only necessary when the next trip of the eVTOL cannot be performed without recharging). We were then interested in how solutions change when the charge constraints are relaxed (thus, allowing a manager to evaluate the tradeoff between the increased throughput and the increased cost of batteries). We ran the same test instances, but changed the charge depletion per time step to zero (effectively removing battery management from the problem). Table 4.3 compares the resulting objective values. We observe that the objective value can increase by 7-10% if we relax battery constraints. We should note that without battery management (i.e., when we can swap batteries as needed instead of waiting to recharge), we can solve problem instances that are (at least) an order of magnitude larger to optimality—-indicating that the battery management constraints make it difficult to solve large-scale instances to optimality.

Finally, we wanted to understand how the throughput varied as the demand changed with a fixed fleet size. When there are 1000 customers, only about 62% of the requests can be accommodated. We decreased the total number of customers from 1000 to 500 in steps of 100. Table 4.4, presents the result of RCG on the first instance of the DC problems, as the total number of customers is varied from 500 to 1000. We see that, though the total throughput decreases as

Problem Instance	Greedy Solution	RCG Solution	Improvement Factor
1	440	648	1.47
2	482	636	1.32
3	461	647	1.40
4	491	661	1.35
5	466	640	1.37

Table 4.2: Results on the DC problem, with 8 ports, 20 eVTOLs, and 1000 customers.

Problem Instance	With Charge Constraints	No Charge Constraints	Improvement Factor
1	648	704	1.09
2	636	690	1.08
3	647	711	1.10
4	661	719	1.09
5	640	687	1.07

Table 4.3: Results on the same DC problems with and without charge constraints.

the total number of customers is reduced, the proportion of satisfied requests increases.

4.7 Conclusions

UAM systems have the promise to improve our daily lives by decreasing the time and cost of moving people and goods in and around cities. They have vast implications for the future of transport, work-life, and urban design. An important ingredient in designing an effective and well-utilized UAM system involves the network logistics.

Number of Customers	Throughput	Percentage
500	404	80.8%
600	479	79.8%
700	532	76.1%
800	579	72.4%
900	622	69.1%
1000	648	64.8%

Table 4.4: Results on the DC problem with 20 eVTOLs and 8 ports.

In this chapter, we consider a problem that would be encountered by a UAM provider in the early stages of the company's development. Specifically, we focus on determining an a priori schedule for the eVTOLs based on the anticipated demand, with the goal of maximizing the throughput. The temporal nature of the demand, time windows for customers, and battery management constraints of the eVTOLs make the UAMP particularly challenging. We develop a three-index, arc-based formulation, routing eVTOLs over a time-expanded network. Due to the computational limitations of the arc-based formulation, we also develop a path-based formulation, and apply a column generation procedure to it. The path-based column generation procedure can be applied in a heuristic manner by (i) sparsifying the time-expanded network, (ii) limiting column generation to the root node in a branch-and-bound scheme, and (iii) applying early termination criteria in the column generation procedure. Our computational experience on a large set of test instances indicates that the path-based approach can identify high-quality solutions for small to large instances.

There are several directions for future research. They include the synchronization of air and ground transportation. In other words, when a customer request for transport arises and it can be served in a multi-modal fashion—it is necessary to coordinate the ground transportation mode with the 'scheduled eVTOL service'. Another issue is the online problem of determining real-time changes to the scheduled eVTOL service to address significant changes or variations in demand for service. Related to the network logistics is the issue of pricing of the eVTOL service.

The UAMP discussed in this chapter is an important first step in developing our understanding, and providing methods for the industry to use. As the industry evolves—with changes in market demands and technology—additional network logistics problems will arise. Their solutions could play an important role in the path towards making eVTOL transport a commercial reality.

Chapter 5: Conclusions

We conclude by looking ahead for each of the three problems we have considered. Specifically, we consider how the respective domains may evolve, as well as how the approaches offered in this dissertation may be applied, adapted, and expanded.

The deregulation of the trucking industry in the 1980s led to the proliferation of LTL transport [97], which, in turn, led to significant cost reductions and improved quality of service. However, inefficiencies remain in the LTL model. The necessity of establishing a comprehensive hub-and-spoke network creates a barrier for entry into the industry, travel along such a network can be circuitous and accident-prone, and the energy consumption of both hubs and long routes is becoming less attractive in light of mounting pressure towards environmentally-friendly policies. Accordingly, the industry is again observing a set of changes, driven by technological development. In particular, the emergence of online platforms, easily accessible to both shippers and carriers, has facilitated direct communication between both parties. This centralization allows for the origins and destinations of shippers to be matched with carriers, facilitating accurate quotes and delivery time estimates. Due to the efficiency of the resulting routes from both an economic and an environmental perspective, the STL model appears likely to grow in prevalence. The approaches developed in this work can be used to tackle the resulting pooling/routing problem. Among the principal unknowns in this space are the costs of routes as the industry evolves

[14]. However, our algorithms, by design, make only modest assumptions with regards to such costs. For example, a pool might break down, meaning a truck encounters service issues mid-route and cannot serve the remaining customers in the pool. If the cost of having to rescue the customers can be included in a given route, the PSA approach we have described is immediately compatible. As STL grows, so may constraints change. We have demonstrated how our work can accommodate such modifications, including the relaxation of LIFO constraints and allowing deadheads. Currently, while trailers capable of side-loading do exist, they are relatively rare. By effectively relaxing LIFO constraints, our work has demonstrated the significant benefits that could come from investment in such trailers. Other constraints, such as separation of freight classes, heterogeneous vehicles and/or time windows, may become necessary to include as the industry expands. In addition, the techniques we have demonstrated can be applied to other routing situations where the terminal locations of a path are factored into its cost.

For smaller deliveries, the most crucial stage is the last mile [98]. Consumers are increasingly shopping online, with, for example, virtual grocery shopping gaining significant momentum in the last decade. Consumer expectations on the timeliness of delivery have also risen, with quality of delivery serving as a deciding element in the customer's overall experience. Furthermore, consumers have demonstrated a willingness to pay a premium for faster deliveries. On the other hand, the last mile is the costliest link in the supply chain. In order to remain competitive, companies sometimes subsidize their deliveries. In light of these trends, innovative solutions abound, including drone-assisted delivery, autonomous vehicles, self-service lockers, and delivery to consumer vehicles. The shuttle-truck synchronization approach outlined in our work falls into this category as a relatively low-tech, minimal infrastructure solution; the improvements of GPS technology can make curbside rendezvous a realistic option. The work in this dissertation serves as a proof of concept for this kind of strategy. However, the setting in which we have applied our work (i.e., shuttles dispatched some fixed length of time after the trucks) may not be optimal. The benefits of truck-shuttle synchronization might be fully realized in settings with multiple depots, and greater flexibility in truck and shuttle dispatch times. For instance, staggered truck dispatches from two depots could allow greater opportunity for interceptions by shuttles, which can start and end their paths at either depot. As delivery expectations become more demanding, and delivery guarantees on the scale of hours become more prevalent, the generalization to the dynamic problem will be necessary. To make deliveries within two hours in a cost-effective manner, the ability to re-route shuttles and trucks mid-trajectory will become an important source of flexibility. However, it may still be the case that a "batching" strategy, where shuttles are dispatched upon the emergence of a given number of LMSLs (implicitly employed in our work), is the most effective approach.

Finally, the future of passenger transit is one of the most important considerations within the Smart City movement. Ground infrastructure has become increasingly congested, leading to increased pollution, car accidents, and portions of lives spent in gridlock. Urban Air Mobility may emerge as a safe, clean alternative. Despite the abundance of regulatory and technological hurdles to be overcome, there is cause to be optimistic about the reality of UAM. Indeed, advancements in electric propulsion and battery technology have led to significant investment from multiple companies racing to be the first to create a viable design. The industry has started to coalesce around key players such as Joby, Volocopter, and Wisk. The work in this dissertation serves to assist a fledgling UAM provider seeking to maximize market share in an intracity mobility setting. However, *inter*-city transport may be the dominant use case. In this setting, the problem statement described in our work is still compatible (merely a modification in distances). Other possibilities include larger cargo deliveries and services such as fighting wildfire, relief, and medevac [99]. In these cases, it appears battery management will remain a decision-driving force, as was the case in our work.

The Smart Cities movement promises a cleaner, safer, and more efficient future. Making it a reality requires not only investment in powerful, widespread, low-latency communication technology, but the ability to use the resulting information effectively. Through mathematical modeling and computational experimentation, this dissertation has worked to serve the latter step.

Appendix A: Additional Material for Chapter 2

A.1 Proofs of Validity

Theorem 2. *Constraints 2.17 are valid for the STLP.*

Proof. Observe that constraints 2.4 imply $y_{ij} \in \{0, 1\}$. We consider cases:

- Case 1: $y_{ij} = 1$. Observe that constraints 2.5, 2.6, and 2.7 imply $y_{ji} = 0$, meaning the fourth term on the left side of 2.17 vanishes.
  - Subcase 1: $\beta_{jk} = 1$. Then, $u_{ik} = 0$, $u_{jk} = 1$, and $u_{ik} - u_{jk} + y_{ij} = 0 - 1 + 1 = 0 = 1 - \beta_{jk}$.
  - Subcase 2: $\beta_{jk} = 0$. Then, $u_{ik} = u_{jk}$, and $(u_{ik} - u_{jk}) + y_{ij} = (0) + 1 = 1 = 1 - \beta_{jk}$.
  - Subcase 3: $\beta_{jk} = -1$. Then, $u_{ik} = 1$, $u_{jk} = 0$, and $u_{ik} - u_{jk} + y_{ij} = 1 - 0 + 1 = 2 = 1 - \beta_{jk}$.
- Case 2: $y_{ij} = 0$. That is, the third term on the left side of 2.17 vanishes.
  - Subcase 1: $y_{ji} = 1$. We must show $u_{ik} - u_{jk} \leq \beta_{ik}$.
    - Subsubcase 1: $\beta_{jk} = 1$. Then, $u_{jk} = 1$, and $u_{ik} - u_{jk} \leq \beta_{ik}$ is only violated if $u_{ik} = 1$ and $\beta_{ik} = -1$, which is impossible.

    * Subsubcase 2: $\beta_{jk} = 0$. Then, if $\beta_{ik} = 1$, then $u_{ik} = 1$, and $u_{jk} = 0$. If $\beta_{ik} = 0$, then $u_{ik} = u_{jk}$. If $\beta_{ik} = -1$, then $u_{jk} = 1$ and $u_{ik} = 0$. In all three cases the constraint is satisfied.
    * Subsubcase 3: $\beta_{jk} = -1$. Then, $u_{ik} = 0$, $u_{jk} = 0$, and $\beta_{ik} = 0$.
  - Subcase 2: $y_{ji} = 0$, meaning the fourth term on the left side of 2.17 vanishes.
    * Subsubcase 1: $\beta_{jk} = 1$. Then, $u_{jk} = 1$, and $u_{ik} - u_{jk} \leq 0 = 1 - \beta_{jk}$.
    * Subsubcase 2: $\beta_{jk} \in \{-1, 0\}$. Then $u_{ik} - u_{jk} \leq 1 - 0 \leq 1 \leq 1 - \beta_{jk}$.

□

Theorem 3. *Constraints 2.18 are valid for the STLP.*

Proof. Note that the second term equals $\beta_{i'k}$, where i' is the node visited by the truck that also visits i whose visit immediately precedes that of i. We consider cases to verify $u_{ik} - \beta_{i'k} \geq \beta_{ik}$.

- Case 1: $\beta_{i'k} = 1$. Then, $\beta_{ik} \in \{-1, 0\}$. If $\beta_{ik} = -1$, then $u_{ik} = 0$. If $\beta_{ik} = 0$, then $u_{ik} = 1$. In either case, the constraint holds.
- Case 2: $\beta_{i'k} = 0$. Then, the constraint is only violated if $u_{ik} = 0$ and $\beta_{ik} = 1$, which is impossible.

□

A.2 Savings Approximation

A key step in the parallel savings algorithm is the calculation of the maximum possible savings associated with combining two routes. Since we compute this value for every pair of routes in a given solution, the performance of the algorithm is highly sensitive to the efficiency of this step. Due to LIFO constraints, the prohibition of intermediate deadheading, and limit of four customers per route, there are at most 120 combinations that need to be evaluated, meaning that computing the savings exactly is inexpensive. However, if LIFO constraints are relaxed, this number rises to 1,776, making exact calculation too slow for the problem sizes we target ($\approx$ 100 customers). We describe here the procedure employed to approximate the maximum savings associated with combining two routes when LIFO constraints are relaxed.

We only compute optimal combinations for 1-routes. Otherwise, we employ an approximation procedure, which appeals to the following insight: due to the iterative nature of the algorithm, the input routes are likely to be "good." That is, the terminal points are likely to be in higher-demand locations, and the order of the nodes visited is likely to be sensible. Rather than throw away this progress, we consider combinations designed to inherit these qualities.

Correspondingly, we evaluate each of the sequence-preserving merges of the two routes. That is, we identify those combinations of the two routes such that the order of the nodes visited in the combination is the same as their order in the original routes. This guarantees that the start/end node in the combination is one of the original start/end nodes. Furthermore, this guarantees that the combination respects precedence constraints, meaning the check for feasibility can ignore this constraint. We note that if the number of stops of the two routes are m and n, respectively, there are $\binom{m+n}{n} = \frac{(m+n)!}{m!n!}$ such combinations. Thus, if a route can serve at most four customers, there are at most $\frac{8!}{4!4!} = 70$ such sequences to evaluate for each combination of two routes. Because the best possible combination of two routes may not be among the above described merges, we also consider routes generated in a greedy fashion. Specifically, for each pickup destination in the original routes, we consider the unique route that emerges from iteratively adding the closest feasible next stop. That is, we identify the greedy route starting from each of the possible starting points. The lowest-cost route generated by either of the procedures is used to calculate savings. Furthermore, if the corresponding arc is included in the matching, the original routes are replaced with the generated route.

Relaxing the deadhead constraints also increases the number of solutions. In particular, if the total number of customers a route can serve is four, there are at most 312 combinations if deadhead constraints are relaxed. We modify the `GetSavings` algorithm in the following way. If the total number of customers served by the input routes is three or fewer, we compute the optimal combination. If the total number of customers is four, we compute the optimal sequence-preserving merge. If both LIFO and deadheading constraints are relaxed, we only consider sequence-preserving merges.

Appendix B: Additional Material for Chapter 4

B.1 Modeling Extensions

The three-index formulation described in the chapter can be extended to incorporate a variety of realistic modeling goals. We discuss such considerations in this section and the ways in which the model can be adapted to include them.

B.1.0.1 Multiple Port Allocation

In our work, we assume a single customer-port allocation. That is, we assume each customer has a unique origin and destination port, when in reality there may be a choice of such ports available. For instance, if the customer's final destination is sufficiently close to two ports, a customer may accept flights to either one.

We extend our definitions in the following way. Suppose there are L possible origin-destination pairs, each with possibly different time windows. We define, for each customer, a set of origin-destination port pairs $\{(o_c^j, d_c^j)\}_{l=1}^L$ and a corresponding set of discrete time windows $\{T_c^j\}_{l=1}^L$. We extend $\mathcal{F}(c)$ to include each of the possible flight arcs for customer c. This will be the collection of arcs $(i,j) \in B$, for which there exists an $l \in \{1,\ldots,L\}$ such that $p_i = o_c^l$, $p_j = d_c^l$, and $t_i \in T_c^l$. Besides this generalization of $\mathcal{F}(c)$, nothing else needs to be changed in

MIP3 to account for multiple port allocation.

B.1.0.2 Soft Time Windows

In many vehicle routing applications involving time windows, a popular extension is the inclusion of ‘soft’ time windows. The idea is to relax each time window constraint and instead penalize its violation in the objective function. This relaxation may be desirable in our setting, where a UAM provider may accept a small loss in quality of service in exchange for increased market share.

Such an extension can be achieved in the following way. For each customer $c \in C$, and for each $(i, j) \in \mathcal{F}(c)$, we define a coefficient $w_{ij}^c \in \mathbb{R}$. This coefficient reflects the value of servicing customer c by flying him along arc (i, j). The coefficients can be determined using any model desired. For instance, if $T_c = \{t_c^a, \ldots, t_c^b\}$ is the contiguous, discrete time window of customer c, one simple such definition is:

$$w_{ij}^c = \frac{t_c^b - t_i}{t_c^b - t_c^a} \qquad \forall (i, j) \in \mathcal{F}(c),$$

which linearly decays from 1 to 0 the later the departure time, t_i. The objective function is then modified to

$$\max \sum_r \sum_{\{(i,j) \in \mathcal{F}(r)\}} w_{ij}^c y_{ij}^c.$$

B.1.0.3 Customer Groups

The three-index formulation assumes each customer is traveling individually, when, in reality, passengers may wish to travel together. Groups of customers can be included in the formulation by simply weighting the objective value of the particular customer (now customer group), c, by the number of customers in the group, n_c, changing the objective to

$$\max \sum_{c \in C} n_c \sum_{\{(i,j) \in \mathcal{F}(c)\}} y_{ij}^c$$

and changing the passenger flow equation to

$$\sum_{\{c|(i,j) \in \mathcal{F}(c)\}} n_c y_{ij}^c \leq \sum_k S x_{ij}^k \quad \forall (i,j) \in \mathcal{F},$$

to account for the increase in load factor.

B.1.0.4 Battery Swaps

The model above assumes charge levels evolve according to e_{ij}, defined for each $(i,j) \in B$. Each arc $(i,j) \in B$ corresponds to a possible transition an eVTOL may make in the time expanded network. If $p_i \neq p_j$, then it must be the case that $t_i + t_{p_i,p_j} = t_j$ (that is, the time at node j must be equal to the time at node i, plus the travel time between the two nodes). We allow for eVTOLs to remain at a port by defining the artificial travel time $t_{ii} = 1$ for each $i \in N$. In the case $p_i \neq p_j$, $e_{ij} < 0$, as an eVTOL is depleting its charge level by flying from one port to another. In the case $p_i = p_j$, $e_{ij} \geq 0$, corresponding to a battery charge increase from recharging. As such arcs correspond to a single time step forward, e_{ij} is specifically the charge increase possible in one time step. This can correspond to a recharging policy or a battery swap policy. In the battery swap policy, $e_{ij} = Q_+$, fully replenishing the charge level.

However, it may be the case that the battery swap process takes more than a single time step. We can account for this by adapting our network in the following way. Suppose a battery swap takes X time steps. For each $i \in M$ such that $t_i + X \in T$, add the arc (i, j) to B, where $p_j = p_i$ and $t_j = t_i + X$. These arcs correspond to remaining at a port for X time steps. Then, for each of these arcs, define $e_{ij} = Q_+$. This will allow the possibility of an eVTOL swapping its battery, with such a choice preventing take-offs during the battery swap period. One can include battery swap and recharging options simultaneously.

B.2 Two-index Heuristic

The charge constraints significantly complicate the problem, and, therefore, limit the problem sizes that can be solved within a reasonable computation time by an MIP solver. In this section, we present an MIP-based heuristic to identify feasible solutions for large problem sizes. We use a two-index formulation of the UAMP with charge constraints relaxed, and then use the output of such a formulation to dramatically sparsify the network prior to re-solving with the three-index formulation.

The charge-relaxed two-index formulation is constructed by aggregating the eVTOL flow variables from MIP3, $x_{ij} = \sum_{k \in K} x_{ij}^k$. The integer-valued variable x_{ij} thus corresponds to the number of eVTOLs traveling along arc (i, j). We have the resulting integer program (MIP2):

$$\max \sum_{c \in C} \sum_{\{(i,j) \in \mathcal{F}(c)\}} y_{ij}^c \tag{B.1}$$

$$\text{s.t.} \quad \sum_{j \in \delta^+(i)} x_{ij} - \sum_{j \in \delta^-(i)} x_{ji} = \begin{cases} |K| & i = s \\ 0 & i \in M \backslash \{s, f\} \\ -|K| & i = f \end{cases} \qquad \forall i \in M \cup \{s, f\} \tag{B.2}$$

$$\sum_{\{c | (i,j) \in \mathcal{F}(c)\}} y_{ij}^c \leq S x_{ij} \qquad \forall (i,j) \in \mathcal{F} \tag{B.3}$$

$$\sum_{(i,j) \in \mathcal{F}(c)} y_{ij}^c \leq 1 \qquad \forall c \in C \tag{B.4}$$

$$x_{ij} \in \mathbb{Z} \qquad \forall i, j \in M \tag{B.5}$$

$$y_{ij}^c \in \{0, 1\} \qquad \forall (i,j) \in \mathcal{F}(r), c \in C \tag{B.6}$$

The objective B.1 maximizes the number of customers that are served. Constraints B.2 ensure continuity of flow throughout the network. Constraints B.3 limit the number of customers flying along an arc by the number of eVTOLs flying along the same arc multiplied by the eVTOL capacity. Constraints B.4 ensure that each customer is served at most once. Constraints B.5 ensure integer-valued eVTOL variables, and B.6 ensure binary-valued passenger flow variables. By both relaxing charge-constraints and aggregating eVTOL flow variables, MIP2 is extremely efficient, running within seconds on large (e.g., $1000+$ customer) problem sizes.

Let R-MIP3 denote MIP3 without the charge constraints. It is clear that any feasible solution of R-MIP3 can be converted into a feasible solution of MIP2. It is straightforward to show that any feasible solution of MIP2 can be converted into a solution of R-MIP3 by disaggregating the integer-valued flow variables x_{ij}. That is, we can identify $x_{ij}^k \in \{0, 1\}$ for all $(i, j) \in B$ and $k \in K$ such that $\sum_k x_{ij}^k = x_{ij}$ and the flow equations hold. Thus, MIP2 provides a set of K throughput-maximizing paths through the network, which may not satisfy charge constraints.

Our sparsification heuristic appeals to the efficiency of MIP2, as well as the throughput-maximizing quality of the arcs in a given solution. The process is as follows. We first solve MIP2, and obtain an optimal solution $x^* = \{x_{ij}^*\}_{(i,j)\in B}$. We then construct a sparsified network B^0. An arc $(i, j) \in B$ is included in B^0 if and only if $x_{ij}^* > 0$ or $p_i = p_j$ and $t_j = t_i + 1$. That is, we only keep the arcs from the solution of MIP2 as well as the arcs corresponding to remaining at a port for consecutive time steps. Keeping these latter arcs ensures charge feasibility. MIP3 can be run on this dramatically sparsified network. We refer to the process of sparsifying in this fashion, and then running MIP3 on the sparsified network as MIP2⇒3.

This heuristic naturally can lead to suboptimal solutions, as the only customer-serving paths in the problem are those produced by MIP2, which may be charge-infeasible. This can be mitigated by, instead of running MIP2 once, running it multiple times, each time cutting off the previously obtained paths. Then, each of the arcs from all the MIP2 runs are included in B, allowing greater system flexibility. However, a balance must be sought, as the more arcs included in B, the longer the computation time.

B.2.0.1 Computational Performance

We evaluated the performance of the two-index heuristic presented in Section 4.4.3, which dramatically reduces the size of the flow network by using a two-index formulation without charge constraints to select candidate arcs. We ran the sparsification procedure on the data,

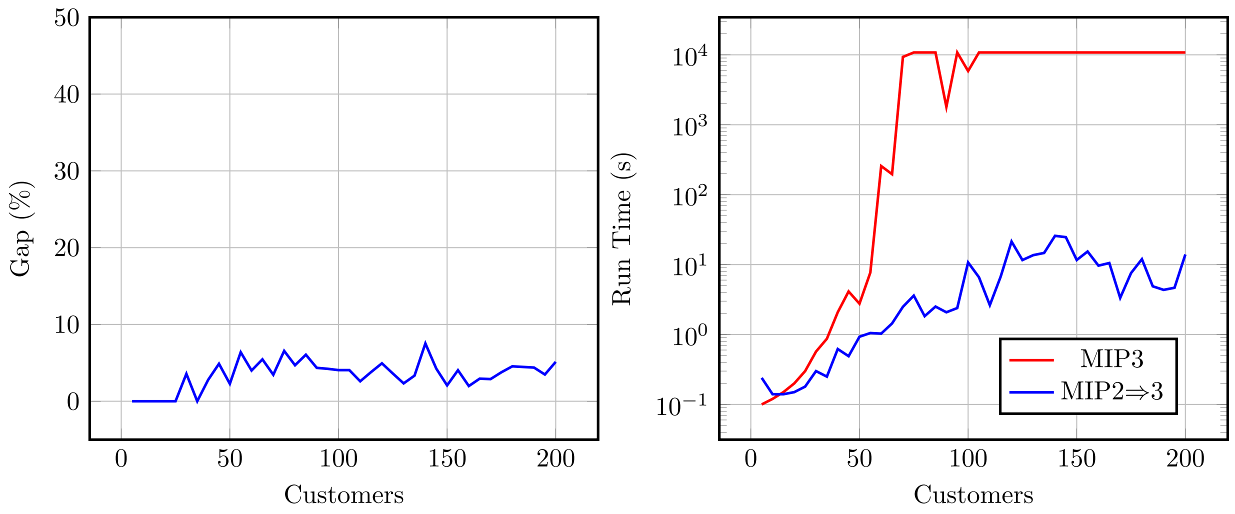


Figure B.1: Gap of MIP2⇒3 and run times of MIP2⇒3 and MIP3 on Problem Set A.

and then applied MIP3 to identify the heuristic solution. We call this procedure ‘Two-Index with Repair’, and denote it MIP2⇒3.

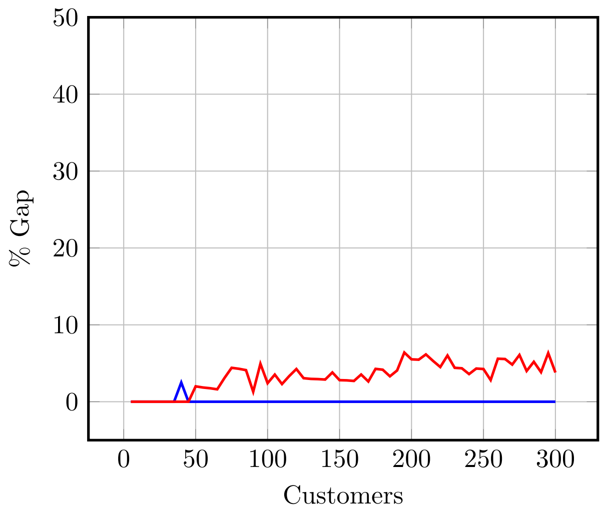
In Figure B.1, the performance on Problem Set A of MIP2⇒3 is compared to that of the exact MIP3. We observe the solution quality is always within 8% of the optimal solution (or the best known solution, if the optimal solution is unavailable). Meanwhile, the run time of MIP2⇒3 is quite insensitive to increases in the number of customers, whereas MIP3 explodes in run time.

However, MIP2⇒3 falls short when compared to the path-based heuristics. In Figure B.2, MIP2⇒3 is compared to RCG, using Problem Set B. We see that (with one exception), RCG produces better solutions in less time, particularly as the number of customers increases.

B.3 Sparsification

We may prove that, with certain assumptions, the sparsification procedure does not cut off the optimal solution.

Theorem 4. *Suppose $Q_+ = \infty$, the distances between ports satisfy the triangle inequality, and*



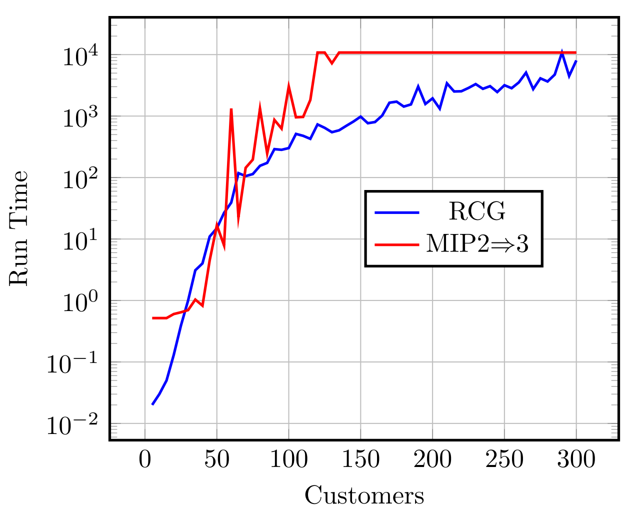


Figure B.2: Gap of MIP2⇒3 and run times of MIP2⇒3 and RCG on Problem Set B.

that the change in charge level due to recharging is convex with respect to the current charge level. Then, the optimal solution to the UAMP has the same objective value as the optimal solution to the UAMP restricted to the stay-put arcs, anticipatory arcs, and flight arcs.

Proof. Let $\mathbf{r} = \{r_1, \ldots, r_n\}$ be an optimal path for the UAMP, where $r_i \in M \; \forall i \in \{1, \ldots, n\}$, and $(r_i, r_{i+1}) \in B \; \forall i \in \{1, \ldots, n-1\}$. We shall construct a charge-feasible path $\mathbf{r'} = \{r'_1, \ldots, r'_m\}$, where $r'_i \in M$ for all $i \in \{1, \ldots, n\}$, that serves the same customers as $\mathbf{r}$, and is such that $(r'_i, r'_{i+1}) \in B'$ for all $i \in \{1, \ldots, n-1\}$.

The construction is as follows. Initialize $\mathbf{r'} = \{r_1\}$. Starting from $i = 1$, we determine if $(r_i, r_{i+1}) \in B'$. If so, append r_{i+1} to $\mathbf{r'}$. Suppose $(r_i, r_{i+1}) \notin B'$. Since $(r_i, r_{i+1}) \notin B'$, the arc is neither a flight arc, nor a stay put arc, nor an anticipatory arc. Since the distances between ports satisfy the triangle inequality, we may assume, without loss of generality, that there is some $k \in \mathbb{Z}^+$ such that $\{(r_{i+1}, r_{i+2}), \ldots, (r_{i+k}, r_{i+k+1})\}$ is a sequence of stay put arcs, and that (r_{i+k}, r_{i+k+1}) is a flight arc. Let $r_i^j \in M$ denote the node satisfying $p_{r_i^j} = p_{r_i}$ and $t_{r_i^j} = t_{r_i} + j$. That is, r_i^j is the port-time node at the same port as r_i, j time steps ahead. We may observe that rather than following the sequence $\{r_i, r_{i+1}, r_{i+2}, \ldots, r_{i+k-1}, r_{i+k}\}$, we may follow the sequence $\{r_i, r_i^1, r_i^2, \ldots, r_i^k, r_{i+k}\}$, consisting of only stay put arcs and an anticipatory arc. Since the charge level is convex with respect to the current charge level, and since $Q_+ = \infty$, the charge level upon arrival at r_{i+k} along the second sequence will be no less than along the first sequence. We therefore can append the nodes $r_i^1, \ldots, r_i^k, r_{i+k}, r_{i+k+1}$ to $\mathbf{r}'$, and continue from $i = i + k + 1$. We continue this until reaching $i = n - 1$, in which case $(r_{n-1}, r_n\}$ is, without loss of generality, either a stay put arc or a flight arc, and thus in B', so r_n may be appended to $\mathbf{r}'$. By construction, $\mathbf{r}'$ is charge-feasible, and furthermore, $\mathbf{r}'$ serves the same customers as the path $\mathbf{r}$. □

We may demonstrate we need all anticipatory arcs, rather than just the ‘extreme’ anticipatory arcs, in order for the sparsified network to keep the optimal solution by means of an example instance of the UAMP, visualized in Figure B.3. In the instance, there is a single eVTOL, and three customers. Passenger 1 wishes to depart from Port 1 towards Port 2 at time 0. Passenger 2 wishes to depart from Port 3 towards Port 4 at either time 5, 6, or 7. Passenger 3 wishes to depart from Port 4 towards Port 5 at time 8. Assuming it is charge feasible, the optimal solution is to fly Passenger 1 from Port 1 to Port 2 at time 0 (arriving at time 3), then to immediately dead head from Port 2 to Port 3. From there, we immediately bring Passenger 2 from Port 3 to Port 4 (departing at time 6), and then immediately take Passenger 3 from Port 4 to Port 5. We note that this sequence is the only way to serve each of the customers, and that it requires a non-extreme anticipatory arc (the one connecting Port 2 to Port 3, departing at time 3). Thus we cannot delete such arcs from the network and guarantee that we do not cut off optimal solutions.

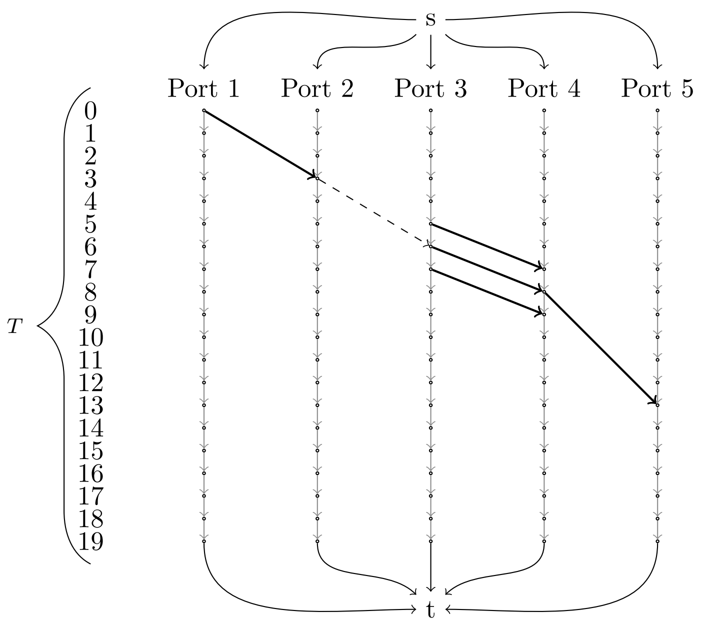


Figure B.3: Demonstration of necessity of all anticipatory arcs.

B.4 Load Factor Analysis

Of particular relevance to a UAM provider may be the average flight occupancy, or load factor. In particular, there may be interest in the proportion of 'deadheads' (flights in the solution in which there are no passengers), flights with a single customer, two customers, and so on. We selected parameters to establish a crowded scenario. In particular, the number of ports was set to four, the number of eVTOLs to three, and the number of time steps to 25. We then were able to quickly solve 500-customer problems. Furthermore, by reducing the time horizon in this fashion, we significantly increase the density of customer flight requests. There are roughly $t_{max} \cdot \binom{|P|}{2}$ possible flight paths, corresponding to every origin-destination pair and every flight time. Thus, when we restrict the number of ports to four and the number of time steps to 25, there are at most 150 possible paths (fewer if it is infeasible to get from any one particular port to another in one time step). If customer requests are distributed uniformly across the possible requests, each with a wait time of three time steps, then varying the number of customers from 50 to 500 corresponds to an expected number of passengers requesting a particular arc varying from roughly 1 to 10.

In this setting, we observed how the load factor of the eVTOLs and the total system throughput varies as we increase the density of customer requests. In Figure B.4, we present the average load factor for four different parameter settings: setting the number of eVTOLs to 3 and 4 and setting the capacities of the eVTOLs to 5 and 6. While all parameter settings appear to have similar results in the extreme cases (highly sparse and highly dense customer requests, respectively), the lower capacity configurations climb slightly faster towards full occupancy.

An important metric for a UAM provider might also be the eVTOL flight time per customer (that is, the total time spent flying divided by the number of customers served). One of the principal costs for a UAM provider might be the recharging of its fleet, and providing service might only be profitable if the flight time per customer is below a certain threshold. In Figure B.5, we plot the average flight time per customer for the four different parameter settings versus the number of customers (corresponding to customer request density). We observe a clear delineation between the different capacity levels: When the capacity is increased from 5 to 6, there is a decrease in flight time per customer. Meanwhile, increasing the fleet size increases the flight time per customer slightly, as the chances of flying at less than full load factor decrease when more eVTOLs are available.

B.5 Tables Corresponding to Computational Results in Chapter 4

Tables B.1 to B.6 present the detailed computational results for the various algorithms discussed in the chapter. For all methods, and all instances, a three hour time limit is imposed.

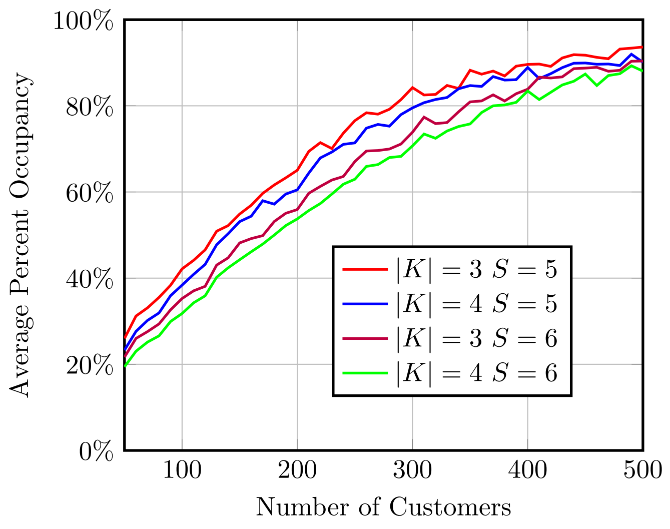


Figure B.4: Occupancy levels for various parameter settings.

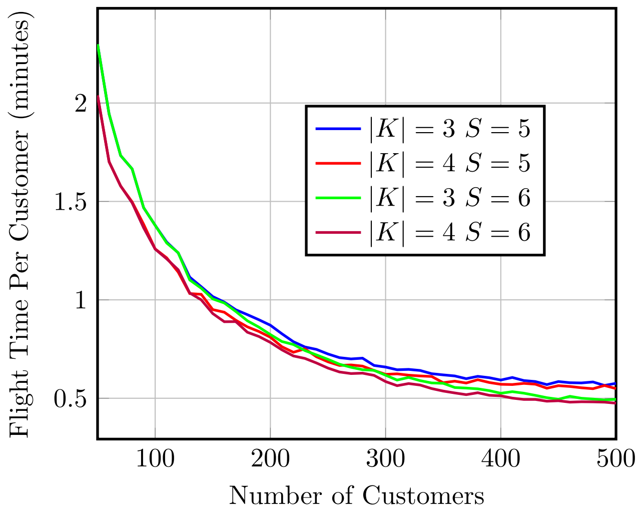


Figure B.5: Average flight times for various parameter settings.

For all tables containing objective values, the best values are bolded. The run times for the various methods on Problem Set A is presented in Table B.1. The corresponding objective values are presented in Table B.2. For the exact methods (MIP3 and B&P), values in parenthesis correspond to the best feasible solution found after three hours. For MIP3, the values in brackets correspond to the best upper bound after three hours. The run times for the better-performing methods on Problem Set B are presented in Table B.3, and the objective values are presented in Table B.4. Finally, the run times for the better-performing methods on Problem Set B* are presented in Table B.5, and the objective values are presented in Table B.6.

$\lvert C\rvert$	MIP3	B&P	MIP2⇒3	RCG	RCG1000	RCG5000	RCG-TH-0.1	RCG-TH-1
5	0.10	0.02	0.24	0.02	0.03	0.02	0.12	0.01
10	0.12	0.03	0.14	0.03	0.03	0.03	0.02	0.02
15	0.15	0.00	0.14	0.01	0.01	0.01	0.00	0.01
20	0.20	0.64	0.15	0.76	0.76	0.81	0.65	0.24
25	0.30	1.62	0.18	0.83	0.86	0.89	0.75	0.45
30	0.57	19.21	0.30	1.95	2.31	2.07	1.70	1.70
35	0.87	7.49	0.25	6.33	5.76	5.54	4.24	3.51
40	2.08	8.57	0.62	7.14	6.39	6.60	5.48	5.44
45	4.13	68.85	0.49	23.21	12.67	21.60	17.69	6.92
50	2.76	10.34	0.93	9.99	10.84	10.52	8.67	5.26
55	7.69	46.44	1.05	26.65	9.01	28.89	23.23	14.52
60	256.53	421.73	1.03	48.99	12.41	53.12	40.79	28.62
65	196.29	510.12	1.44	59.28	11.09	45.23	34.08	19.37
70	9331.85	1237.34	2.49	213.70	13.13	194.60	95.90	28.63
75	-	442.87	3.60	152.88	11.39	97.23	93.33	24.22
80	-	4124.29	1.83	265.44	13.35	183.63	95.45	44.49
85	-	1678.56	2.51	211.46	13.38	112.65	94.23	33.38
90	1802.00	453.70	2.08	390.62	16.90	155.37	89.80	41.40
95	-	4350.80	2.39	565.81	11.06	390.67	114.31	49.59
100	5859.01	1324.05	10.74	678.49	11.00	346.09	174.26	63.68
105	-	-	6.59	350.01	16.51	201.85	151.96	65.25
110	-	2108.75	2.63	758.20	12.54	420.80	171.61	71.99
115	-	2227.74	6.74	472.44	10.45	432.99	187.21	62.81
120	-	4251.86	21.37	450.73	16.59	531.97	389.80	82.36
125	-	1761.92	11.61	1383.51	20.54	591.81	301.59	112.10
130	-	6979.26	13.65	446.69	13.39	595.59	327.11	87.00
135	-	473.77	14.68	416.73	17.55	326.19	471.23	82.25
140	-	4127.90	25.78	1012.70	19.62	411.56	328.45	110.07
145	-	465.69	24.67	702.39	17.28	981.10	894.33	117.06
150	-	3154.59	11.63	434.50	23.00	296.18	404.18	198.66
155	-	518.08	15.39	499.90	17.60	270.52	448.89	184.74
160	-	1892.63	9.66	479.97	18.90	308.77	437.32	108.63
165	-	276.66	10.53	298.51	23.13	358.18	279.36	137.36
170	-	1323.80	3.34	774.97	21.01	381.62	742.45	207.75
175	-	1958.68	7.57	1151.33	20.38	284.00	1081.95	235.68
180	-	-	11.96	1701.53	18.48	200.04	1343.20	425.97
185	-	1461.04	4.89	998.20	23.01	180.84	935.21	332.32
190	-	-	4.35	9308.63	22.61	241.64	8997.49	329.72
195	-	-	4.66	668.65	21.79	144.53	757.74	428.56
200	-	-	13.96	1248.66	21.78	153.82	733.00	298.59

Table B.1: Problem Set A run times.

$\lvert C\rvert$	MIP3	B&P	MIP2⇒3	RCG	RCG1000	RCG5000	RCG-TH-0.1	RCG-TH-1
5	**5**	**5**	**5**	**5**	**5**	**5**	**5**	4
10	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
15	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**
20	**19**	**19**	**19**	**19**	**19**	**19**	**19**	18
25	**24**	**24**	**24**	23	23	23	23	22
30	**28**	**28**	27	27	27	27	27	27
35	**33**	**33**	**33**	**33**	**33**	**33**	**33**	32
40	**36**	**36**	35	**36**	**36**	**36**	**36**	**36**
45	**41**	**41**	39	40	40	40	40	40
50	**44**	**44**	43	**44**	**44**	**44**	**44**	**44**
55	**47**	**47**	44	**47**	43	**47**	**47**	45
60	**50**	**50**	48	**50**	46	**50**	**50**	49
65	**55**	**55**	52	**55**	51	**55**	**55**	54
70	**58**	**58**	56	**58**	53	**58**	**58**	57
75	(61)[62]	**61**	57	**61**	56	**61**	**61**	60
80	(64)[65]	**64**	61	63	57	63	63	61
85	(66)[67]	**66**	62	**66**	58	**66**	**66**	64
90	69	**69**	66	**69**	62	**69**	**69**	68
95	(71)[72]	**71**	68	**71**	61	**71**	**71**	70
100	74	**74**	71	**74**	64	**74**	**74**	73
105	(**74**)[76]	(**74**)	71	**74**	65	**74**	**74**	73
110	(**77**)[79]	**77**	75	**77**	67	**77**	**77**	**77**
115	(77)[83]	**79**	76	**79**	73	**79**	**79**	78
120	(**81**)[85]	**81**	77	**81**	71	**81**	**81**	79
125	(83)[88]	**84**	80	**84**	73	83	83	83
130	(85)[91]	**86**	84	**86**	74	**86**	**86**	85
135	(89)[94]	**90**	87	**90**	76	**90**	**90**	89
140	(91)[97]	**93**	86	**93**	84	**93**	**93**	**93**
145	(93)[99]	**94**	90	**94**	81	**94**	**94**	93
150	(97)[103]	**98**	95	97	82	97	97	97
155	(97)[104]	**100**	95	99	73	99	99	99
160	(**102**)[106]	**102**	99	101	88	101	101	100
165	(101)[106]	**102**	99	**102**	88	101	**102**	101
170	(103)[108]	**104**	101	**104**	79	**104**	**104**	103
175	(105)[110]	**106**	102	**106**	78	**106**	**106**	104
180	(109)[114]	(**110**)	105	**110**	85	109	**110**	109
185	(**112**)[116]	**112**	107	**112**	95	111	**112**	**112**
190	(113)[118]	(**114**)	109	**114**	100	**114**	**114**	**114**
195	(**116**)[121]	(115)	111	115	93	114	115	114
200	(**117**)[124]	(**117**)	111	**117**	97	114	**117**	116

Table B.2: Problem Set A objective values.

$\lvert C \rvert$	MIP2⇒3	RCG	TH-0.1	TH-1	5K	$\lvert C \rvert$	RCG	TH-0.1	TH-1	5K
50	16.71	14.99	14.36	14.90	14.26	180	1426.21	938.63	566.89	132.92
55	7.90	26.63	25.52	15.22	25.71	185	1546.70	964.29	487.61	161.67
60	1329.88	39.16	37.46	27.33	48.65	190	3018.32	1179.87	632.71	155.45
65	22.83	117.97	70.01	34.27	128.95	195	1572.06	1110.85	596.91	198.65
70	143.82	105.54	79.23	43.19	117.59	200	1941.76	1215.70	467.46	206.68
75	195.88	113.87	76.97	48.29	116.46	205	1321.91	1035.66	537.96	171.20
80	1316.59	155.29	95.66	59.01	152.20	210	3407.82	1462.72	794.22	207.87
85	246.78	173.16	121.40	57.93	171.38	215	2511.90	1530.85	679.69	185.31
90	867.10	289.71	177.44	125.28	130.00	220	2530.22	1304.18	605.49	173.17
95	625.37	282.65	190.96	110.72	146.56	225	2866.63	1663.74	705.77	186.45
100	3018.94	301.34	222.02	117.55	188.68	230	3312.65	1868.20	677.83	169.47
105	954.91	512.73	327.45	143.31	130.40	235	2774.25	2252.59	1100.95	210.71
110	968.78	477.10	293.19	139.02	133.44	240	3065.07	2136.53	917.93	200.28
115	1828.50	426.10	348.62	171.46	156.00	245	2458.76	1640.78	701.38	188.20
120	-	731.29	450.96	192.95	136.26	250	3175.04	2112.46	958.49	221.19
125	-	643.29	418.90	197.04	174.08	255	2838.78	1777.94	831.14	229.44
130	7223.39	546.30	437.25	232.38	141.74	260	3525.95	1936.10	861.40	202.58
135	-	586.42	428.78	213.42	122.71	265	5087.25	1471.67	812.28	240.54
140	-	693.43	416.50	250.17	139.83	270	2746.22	2016.43	1048.99	312.02
145	-	814.92	589.67	307.64	109.76	275	4108.26	2077.59	1062.55	267.96
150	-	979.33	634.90	318.43	166.60	280	3650.89	1937.69	998.77	297.42
155	-	764.67	535.62	280.33	122.35	285	4728.99	3585.57	1453.53	247.58
160	-	800.13	606.47	281.82	140.54	290	10775.53	3606.87	1495.18	277.19
165	-	1021.26	786.12	375.09	121.08	295	4486.52	2906.29	1257.12	224.44
170	-	1649.21	967.40	452.61	131.14	300	8045.45	3939.81	1477.68	269.14
175	-	1716.01	1026.88	460.01	189.94					

Table B.3: Problem Set B run times.

$\lvert C\rvert$	MIP2⇒3	RCG	TH-0.1	TH-1	5K	$\lvert C\rvert$	MIP2⇒3	RCG	TH-0.1	TH-1	5K
50	49	**50**	**50**	**50**	**50**	180	115	**120**	119	117	112
55	53	**54**	**54**	52	**54**	185	117	**121**	**121**	119	114
60	56	**57**	**57**	56	**57**	190	118	**123**	**123**	122	112
65	61	**62**	**62**	60	**62**	195	117	**125**	**125**	123	114
70	63	**65**	**65**	63	**65**	200	120	**127**	**127**	124	118
75	65	**68**	**68**	66	**68**	205	121	**128**	**128**	127	117
80	67	**70**	**70**	69	**70**	210	122	**130**	**130**	129	120
85	70	**73**	**73**	72	**73**	215	125	**132**	**132**	130	121
90	76	**77**	**77**	75	**77**	220	127	**133**	**133**	133	122
95	77	**81**	80	79	80	225	125	**133**	**133**	132	120
100	81	**83**	**83**	80	82	230	130	**136**	**136**	135	123
105	82	**85**	**85**	84	84	235	132	**138**	**138**	137	123
110	85	**87**	**87**	85	86	240	134	**139**	**139**	137	127
115	87	**90**	**90**	88	88	245	133	**139**	**139**	136	128
120	90	**94**	**94**	92	91	250	135	**141**	**141**	139	129
125	95	**98**	**98**	96	96	255	138	**142**	**142**	140	129
130	98	**101**	**101**	99	97	260	135	**143**	**143**	141	128
135	99	**102**	**102**	99	96	265	136	**144**	**144**	141	136
140	101	**104**	**104**	103	99	270	138	**145**	**145**	143	129
145	101	**105**	**105**	104	99	275	139	**148**	**148**	146	137
150	104	**107**	106	105	101	280	144	**150**	**150**	149	138
155	105	**108**	**108**	106	104	285	146	**154**	**154**	151	141
160	108	**111**	**111**	108	107	290	150	**156**	**156**	153	144
165	109	**113**	112	111	106	295	148	**158**	157	155	136
170	111	**114**	**114**	112	103	300	152	**158**	**158**	156	146
175	112	**117**	**117**	114	110						

Table B.4: Problem Set B objective values.

$\lvert C\rvert$	RCG	RCG-TH-1	RCG5000	RCG-1hr
50	7.47	5.40	7.34	7.47
100	82.71	36.68	77.36	82.71
150	175.50	93.45	84.80	175.50
200	260.35	153.83	110.27	260.35
250	223.84	180.04	113.25	223.84
300	517.36	216.42	126.83	517.36
350	783.61	368.97	125.93	783.61
400	1348.88	631.26	145.69	1348.88
450	1977.14	534.88	141.25	1977.14
500	2489.50	939.86	157.97	2489.50
550	2553.20	1129.97	180.98	2553.20
600	1820.28	1301.80	202.19	1820.28
650	2111.24	1242.99	187.52	2111.24
700	7551.02	1975.04	236.80	3652.00
750	13575.44	2235.51	253.19	3662.51
800	8595.37	1972.04	276.69	3625.09
850	10262.08	3150.56	268.09	3954.77
900	8919.09	3861.25	213.83	3813.95

Table B.5: Problem Set B* run times.

$\|C\|$	RCG	RCG-TH-1	RCG5000	RCG-1hr
50	**48**	**48**	**48**	**48**
100	**81**	79	**81**	**81**
150	**103**	102	101	**103**
200	**121**	**121**	120	**121**
250	**133**	**133**	130	**133**
300	**151**	**151**	146	**151**
350	**169**	**169**	161	**169**
400	**186**	185	169	**186**
450	**203**	201	186	**203**
500	**213**	211	192	**213**
550	**228**	226	206	**228**
600	**241**	**241**	221	**241**
650	**252**	250	242	**252**
700	**267**	265	240	**267**
750	**281**	279	252	**281**
800	**290**	**290**	257	**290**
850	**303**	302	271	**303**
900	**320**	318	271	318

Table B.6: Problem Set B* objective values.

Bibliography

[1] G. B. Dantzig and J. H. Ramser. The Truck Dispatching Problem. *Management Science*, 6 (1):80–91, 1959. URL http://www.jstor.org/stable/2627477.

[2] Richard Karp. Reducibility among combinatorial problems. In Raymond Miller, James Thatcher, and Jean Bohlinger, editors, *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*, The IBM Research Symposia Series, pages 85–103. Springer US, Boston, MA, 1972. ISBN 978-1-4684-2001-2. doi: 10.1007/978-1-4684-2001-2_9. URL https://doi.org/10.1007/978-1-4684-2001-2_9.

[3] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40 (2):342–354, April 1992. ISSN 0030-364X. doi: 10.1287/opre.40.2.342. URL https://pubsonline.informs.org/doi/abs/10.1287/opre.40.2.342. Publisher: INFORMS.

[4] Xingyin Wang. Vehicle Routing Problems that Minimize the Completion Time: Heuristics, Worst-Case Analyses, and Computational Results. 2016. doi: 10.13016/M27F7Z. URL https://drum.lib.umd.edu/handle/1903/18710. Accepted: 2016-09-08T05:34:44Z.

[5] Harilaos N. Psaraftis. Dynamic vehicle routing: Status and prospects. *Annals of Operations Research*, 61(1):143–164, December 1995. ISSN 1572-9338. doi: 10.1007/BF02098286. URL https://doi.org/10.1007/BF02098286.

[6] Lawrence Bodin. A taxonomic structure for vehicle routing and scheduling problems. *Computers & Urban Society*, 1(1):11–29, January 1975. ISSN 0305-7097. doi: 10.1016/0305-7097(75)90003-4. URL https://www.sciencedirect.com/science/article/pii/0305709775900034.

[7] Lawrence Bodin and Bruce Golden. Classification in vehicle routing and scheduling. *Networks*, 11(2):97–108, 1981. ISSN 1097-0037. doi: 10.1002/net.3230110204. URL

https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230110204. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230110204.

[8] M. Desrochers, J. K. Lenstra, and M. W. P. Savelsbergh. A classification scheme for vehicle routing and scheduling problems. *European Journal of Operational Research*, 46(3):322–332, June 1990. ISSN 0377-2217. doi: 10.1016/0377-2217(90)90007-X. URL https://www.sciencedirect.com/science/article/pii/037722179090007X.

[9] Burak Eksioglu, Arif Volkan Vural, and Arnold Reisman. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4):1472–1483, November 2009. ISSN 0360-8352. doi: 10.1016/j.cie.2009.05.009. URL https://www.sciencedirect.com/science/article/pii/S0360835209001405.

[10] Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon. *Column Generation*. Springer Science & Business Media, March 2006. ISBN 978-0-387-25486-9.

[11] Bhagya Nathali Silva, Murad Khan, and Kijun Han. Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities. *Sustainable Cities and Society*, 38:697–713, April 2018. ISSN 2210-6707. doi: 10.1016/j.scs.2018.01.053. URL https://www.sciencedirect.com/science/article/pii/S2210670717311125.

[12] United Nations, Department of Economic and Social Affairs, and Population Division. *World urbanization prospects: the 2018 revision*. 2019. ISBN 978-92-1-148319-2. OCLC: 1120698127.

[13] Alan Hooper and Dan Murray. E-Commerce Impacts on the Trucking Industry. Technical report, American Transportation Research Institute, February 2019.

[14] Evren Özkaya, Pınar Keskinocak, V. Roshan Joseph, and Ryan Weight. Estimating and benchmarking Less-than-Truckload market rates. *Transportation Research Part E: Logistics and Transportation Review*, 46(5):667–682, September 2010. ISSN 1366-5545. doi: 10.1016/j.tre.2009.09.004. URL https://www.sciencedirect.com/science/article/pii/S1366554509001240.

[15] Troy Segal. Less-Than-Truckload (LTL), June 2021. URL https://www.investopedia.com/terms/l/lessthantruckload.asp.

[16] Alan Erera, Michael Hewitt, Martin Savelsbergh, and Yang Zhang. Creating schedules and computing operating costs for LTL load plans. *Computers & Operations Research*, 40(3):691–702, March 2013. ISSN 0305-0548. doi: 10.1016/j.cor.2011.10.001. URL https://www.sciencedirect.com/science/article/pii/S0305054811002887.

[17] Ahmad Jarrah, Ellis Johnson, and Lucas Neubert. Large-Scale, Less-than-Truckload Service Network Design. *Operations Research*, 57(3):609–625, June 2009. ISSN 0030-364X. doi: 10.1287/opre.1080.0587. URL http://pubsonline.informs.org/doi/abs/10.1287/opre.1080.0587. Publisher: INFORMS.

[18] Teodor Gabriel Crainic. A Survey of Optimization Models for Long-Haul Freight Transportation. Technical report, Center for Research on Transportation, 2002.

[19] Ian Herszterg, Yassine Ridouane, Natashia Boland, Alan Erera, and Martin Savelsbergh. Near real-time loadplan adjustments for less-than-truckload carriers. *European Journal of Operational Research*, December 2021. ISSN 0377-2217. doi: 10.1016/j.ejor.2021.11.044. URL https://www.sciencedirect.com/science/article/pii/S0377221721009954.

[20] Warren Powell and Yosef Sheffi. The load planning problem of motor carriers: Problem description and a proposed solution approach. *Transportation Research Part A: General*, 17(6):471–480, November 1983. ISSN 0191-2607. doi: 10.1016/0191-2607(83)90167-X. URL https://www.sciencedirect.com/science/article/pii/019126078390167X.

[21] Naoto Katamaya and Shigeru Yurimoto. The Load Planning Problem for Less-Than-Truckload Motor: Carriers and a Solution Approach. In *Developments in Logistics and Supply Chain Management*, pages 240–249. Palgrave Macmillan, January 2002. ISBN 978-1-349-55848-3.

[22] Alan Erera, Michael Hewitt, Martin Savelsbergh, and Yang Zhang. Improved Load Plan Design Through Integer Programming Based Local Search. *Transportation Science*, 47(3): 412–427, August 2013. ISSN 0041-1655. doi: 10.1287/trsc.1120.0441. URL http://pubsonline.informs.org/doi/abs/10.1287/trsc.1120.0441. Publisher: INFORMS.

[23] L. Barcos, V. Rodríguez, M. J. Álvarez, and F. Robusté. Routing design for less-than-truckload motor carriers using Ant Colony Optimization. *Transportation Research Part E: Logistics and Transportation Review*, 46(3):367–383, May 2010. ISSN 1366-5545. doi: 10.1016/j.tre.2009.11.006. URL https://www.sciencedirect.com/science/article/pii/S1366554509001392.

[24] Srinivas Subramanya Tamvada, Bahareh Mansouri, Elkafi Hassini, and Theodore Pribytkov. An integer programming model and directed Steiner-forest based heuristic for routing less-than-truckload freight. *International Journal of Production Economics*, 232:107925, February 2021. ISSN 0925-5273. doi: 10.1016/j.ijpe.2020.107925. URL https://www.sciencedirect.com/science/article/pii/S0925527320302802.

[25] Sin C. Ho, W.Y. Szeto, Yong-Hong Kuo, Janny M.Y. Leung, Matthew Petering, and Terence W.H. Tou. A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological*, 111:395–421, May 2018. ISSN 01912615. doi: 10.1016/j.trb.2018.02.001. URL https://linkinghub.elsevier.com/retrieve/pii/S0191261517304484.

[26] Sophie Parragh, Karl Doerner, and Richard Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(2):81–117, June 2008. ISSN 1614-631X. doi: 10.1007/s11301-008-0036-4. URL https://doi.org/10.1007/s11301-008-0036-4.

[27] Marilène Cherkesly, Guy Desaulniers, and Gilbert Laporte. Branch-Price-and-Cut Algorithms for the Pickup and Delivery Problem with Time Windows and Last-in-First-Out Loading. *Transportation Science*, 49(4):752–766, November 2015. ISSN 0041-1655. doi: 10.1287/trsc.2014.0535. URL `http://pubsonline.informs.org/doi/10.1287/trsc.2014.0535`. Publisher: INFORMS.

[28] Jean-François Cordeau, Manuel Iori, Gilbert Laporte, and Juan José Salazar González. A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with LIFO loading. *Networks*, 55(1):46–59, 2010. ISSN 1097-0037. doi: 10.1002/net.20312. URL `http://onlinelibrary.wiley.com/doi/abs/10.1002/net.20312`. _-eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.20312.

[29] Enrique Benavent, Mercedes Landete, Enrique Mota, and Gregorio Tirado. The multiple vehicle pickup and delivery problem with LIFO constraints. *European Journal of Operational Research*, 243(3):752–762, June 2015. ISSN 03772217. doi: 10.1016/j.ejor.2014.12.029. URL `https://linkinghub.elsevier.com/retrieve/pii/S0377221714010479`.

[30] Andriansyah, Nissa Prasanti, and Prima Denny Sentia. Pickup and delivery problem with LIFO, time duration, and limited vehicle number. *MATEC Web of Conferences*, 204: 07003, 2018. ISSN 2261-236X. doi: 10.1051/matecconf/201820407003. URL `https://www.matec-conferences.org/10.1051/matecconf/201820407003`.

[31] L. Cassani and Giovanni Righini. Heuristic Algorithms for the TSP with rear-loading. Lecce, Italy, September 2004. URL `http://optlab.dti.unimi.it/Papers/Cassani.pdf`.

[32] Francesco Carrabs, Jean-françois Cordeau, and Gilbert Laporte. Variable neighborhood search for the pickup and delivery traveling salesman problem with lifo loading. *INFORMS Journal on Computing*, 19(4):618–632, 2007.

[33] D. Tu, Songshan Guo, Hu Qin, Wee-Chong Oon, and A. Lim. The Tree Representation of Feasible Solutions for the TSP with Pickup and Delivery and LIFO Loading. In *AAAI*, 2010.

[34] Xiang Gao, Andrew Lim, Hu Qin, and Wenbin Zhu. Multiple Pickup and Delivery TSP with LIFO and Distance Constraints: A VNS Approach. In Kishan G. Mehrotra, Chilukuri K. Mohan, Jae C. Oh, Pramod K. Varshney, and Moonis Ali, editors, *Modern Approaches in Applied Intelligence*, Lecture Notes in Computer Science, pages 193–202, Berlin, Heidelberg, 2011. Springer. ISBN 978-3-642-21827-9. doi: 10.1007/978-3-642-21827-9_20.

[35] Martin Desrochers and T. W. Verhoog. A Matching Based Savings Algorithm for the Vehicle Routing Problem, 1989. URL `/paper/A-Matching-Based-Savings-Algorithm-for-the-Vehicle-Desrochers-Verhoog/1cca59362704ba5397b92818df3e5d2389fb6994`.

[36] Kemal Altinkemer and Bezalel Gavish. Parallel Savings Based Heuristics for the Delivery Problem. *Operations Research*, 39(3):456–469, June 1991. ISSN 0030-364X. doi: 10.1287/opre.39.3.456. URL `https://pubsonline.informs.org/doi/10.1287/opre.39.3.456`.

[37] Y. Gajpal and P. Abad. Saving-based algorithms for vehicle routing problem with simultaneous pickup and delivery. *The Journal of the Operational Research Society*, 61 (10):1498–1509, 2010. ISSN 0160-5682. URL `http://www.jstor.org/stable/40802326`. Publisher: Palgrave Macmillan Journals.

[38] C. E. Miller, Albert Tucker, and Richard Zemlin. Integer Programming Formulation of Traveling Salesman Problems. *Journal of the ACM*, 7(4):326–329, October 1960. ISSN 0004-5411. doi: 10.1145/321043.321046. URL `http://doi.org/10.1145/321043.321046`.

[39] Stefan Ropke and Jean-François Cordeau. Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, 43(3):267–286, August 2009. ISSN 0041-1655, 1526-5447. doi: 10.1287/trsc.1090.0272. URL `http://pubsonline.informs.org/doi/abs/10.1287/trsc.1090.0272`.

[40] Richard Stanley. *Enumerative Combinatorics*, volume 2 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, January 1999. ISBN 978-0-521-78987-5.

[41] Aric Hagberg, Daniel Schult, and Pieter Swart. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science conference (SciPy 2008)*, pages 11–15, 2008.

[42] Jack Edmonds. Paths, Trees, and Flowers. *Canadian Journal of Mathematics*, 17: 449–467, 1965. ISSN 0008-414X, 1496-4279. doi: 10.4153/CJM-1965-045-4. URL `https://www.cambridge.org/core/journals/canadian-journal-of-mathematics/article/paths-trees-and-flowers/08B492B72322C4130AE800C0610E0E21`. Publisher: Cambridge University Press.

[43] U.S. Census. Quarterly retail e-commerce sales 3rd quarter 2020, November 2020. URL `https://www.census.gov/retail/mrts/www/data/pdf/ec_current.pdf?`

[44] Alina Selyukh. Optimized prime: how AI and anticipation power amazon's 1-hour deliveries, November 2018. URL `https://www.npr.org/2018/11/21/660168325/optimized-prime-how-ai-and-anticipation-power-amazons-1-hour-deliveries`.

[45] Nils Boysen, Stefan Fedtke, and Stefan Schwerdfeger. Last-mile delivery concepts: a survey from an operational research perspective. *OR Spectrum*, 43(1):1–58, September 2020. ISSN 0171-6468, 1436-6304.

[46] Kees Jacobs, Shannon Warner, Marc Rietra, Lindsey Mazza, Jerome Buvat, Amol Khadikar, Sumit Cherian, and Yashwardhan Khemka. The last-mile delivery challenge. Technical report, 2019. URL `https://www.capgemini.com/wp-content/uploads/2019/01/Report-Digital-%E2%80%93-Last-Mile-Delivery-Challenge1.pdf`.

[47] Michael Drexl. Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints. *Transportation Science*, 46(3):297–316, 2012. ISSN 0041-1655.

[48] Chris Groër, Bruce Golden, and Edward Wasil. The Consistent Vehicle Routing Problem. *Manufacturing & Service Operations Management*, 11(4):630–643, 2009.

[49] Michael Drexl. Applications of the vehicle routing problem with trailers and transshipments. *European Journal of Operational Research*, 227(2):275–283, June 2013. ISSN 0377-2217.

[50] Martin Fink, Guy Desaulniers, Markus Frey, Ferdinand Kiermaier, Rainer Kolisch, and François Soumis. Column generation for vehicle routing problems with multiple synchronization constraints. *European Journal of Operational Research*, 272(2):699–711, January 2019. ISSN 03772217.

[51] Cristián Cortés, Martín Matamala, and Claudio Contardo. The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *European Journal of Operational Research*, 200(3):711–724, February 2010. ISSN 0377-2217. doi: 10.1016/j.ejor.2009.01.022.

[52] A. Rais, F. Alvelos, and M.S. Carvalho. New mixed integer-programming model for the pickup-and-delivery problem with transshipment. *European Journal of Operational Research*, 235(3):530–539, June 2014. ISSN 03772217.

[53] Snežana Mitrović-Minić and Gilbert Laporte. The pickup and delivery problem with time windows and transshipment. *INFOR: Information Systems and Operational Research*, 44(3):217–227, August 2006. ISSN 0315-5986.

[54] Renaud Masson, Fabien Lehuédé, and Olivier Péton. An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science*, 47:344–355, August 2013.

[55] Afonso Sampaio, Martin Savelsbergh, Lucas P. Veelenturf, and Tom Van Woensel. Delivery systems with crowd-sourced drivers: a pickup and delivery problem with transfers. *Networks*, 76(2):232–255, 2020. ISSN 1097-0037.

[56] Sam Thangiah, Adel Fergany, and Salman Awan. Real-time split-delivery pickup and delivery time window problems with transfers. *Central European Journal of Operations Research*, 15(4):329–349, November 2007. ISSN 1613-9178.

[57] Marlin Ulmer, Barrett Thomas, and Dirk Mattfeld. Preemptive depot returns for dynamic same-day delivery. *EURO Journal on Transportation and Logistics*, 8(4):327–361, December 2019. ISSN 2192-4384.

[58] Stacy Voccia, Ann Melissa Campbell, and Barrett Thomas. The same-day delivery problem for online purchases. *Transportation Science*, 53(1):167–184, May 2017. ISSN 0041-1655.

[59] Nabila Azi, Michel Gendreau, and Jean-Yves Potvin. A dynamic vehicle routing problem with multiple delivery routes. *Annals of Operations Research*, 199(1):103–112, October 2012. ISSN 1572-9338.

[60] Mathias Klapp, Alan Erera, and Alejandro Toriello. The one-dimensional dynamic dispatch waves problem. *Transportation Science*, 52(2):402–415, May 2016. ISSN 0041-1655.

[61] Cynthia Barnhart, Ellis Johnson, George Nemhauser, Martin Savelsbergh, and Pamela H. Vance. Branch-and-price: column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, June 1998. ISSN 0030-364X.

[62] Ravindra Ahuja, Thomas Magnanti, and J. Orlin. *Network flows - theory, algorithms and applications*. Prentice Hall, 1993.

[63] Keld Helsgaun. An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, October 2000. ISSN 03772217. doi: 10.1016/S0377-2217(99)00284-2. URL `https://linkinghub.elsevier.com/retrieve/pii/S0377221799002842`.

[64] Marius Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, April 1987. ISSN 0030-364X.

[65] Hermann Gehring and Jörg Homberger. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In *Proceedings of EUROGEN99–Short Course on Evolutionary Algorithms in Engineering and Computer Science*, pages 57–64, 1999.

[66] Uber. Fast-Forwarding to a Future of On-Demand Urban Air Transportation. 2016.

[67] Madeline Stone. 'Uber-for-helicopters' startup Blade just raised $38 million — here's what it's like to fly to the Hamptons, 2015. URL `https://www.businessinsider.com/what-blade-the-uber-for-helicopters-is-like-2017-5`. Library Catalog: www.businessinsider.com.

[68] Andrew J. Hawkins. Electric air taxi startup Lilium completes first test of its new five-seater aircraft, May 2019. URL `https://www.theverge.com/2019/5/16/18625088/lilium-jet-test-flight-electric-aircraft-flying-car`. Library Catalog: www.theverge.com.

[69] Jan Hendrik. Pioneering the Urban Air Taxi Revolution. Technical report, Volocopter, 2020. URL `https://press.volocopter.com/images/pdf/Volocopter-WhitePaper-1-0.pdf`.

[70] Huaxiang Xu. The Future of Transportation: White Paper on Urban Air Mobility Systems. Technical report, EHang, 2020. URL https://www.ehang.com/app/en/EHang%20White%20Paper%20on%20Urban%20Air%20Mobility%20Systems.pdf.

[71] M.E. O'Kelly, D. Bryan, D. Skorin-Kapov, and J. Skorin-Kapov. Hub network design with single and multiple allocation: A computational study. *Location Science*, 4(3):125–138, October 1996. ISSN 09668349. doi: 10.1016/S0966-8349(96)00015-0. URL https://linkinghub.elsevier.com/retrieve/pii/S0966834996000150.

[72] Liting Chen, Sebastian Wandelt, Weibin Dai, and Xiaoqian Sun. Scalable vertiport hub location selection for air taxi operations in a metropolitan region. *INFORMS Journal on Computing*, 34(2):834–856, 2022.

[73] E Lim and H Hwang. The selection of vertiport location for on-demand mobility and its application to Seoul metro area. *International Journal of Aeronautical and Space Sciences*, 20(1):260–272, 2019.

[74] Kai Wang, Alexander Jacquillat, and Vikrant Vaze. Vertiport planning for urban aerial mobility: An adaptive discretization approach port placement for uam service in a multimodal transportation network. Technical report, Heinz College of Information Systems and Public Policy, Carnegie Mellon University, 2021.

[75] Zhiqiang Wu and Yu Zhang. Integrated network design and demand forecast for on-demand urban air mobility. *Engineering*, 7:473–487, 2021.

[76] Mercedes Pelegrin and Claudia D'Ambrosio. Aircraft deconfliction via mathematical programming: Review and insights. *Transportation Science*, 56(1):118–140, 2022.

[77] Hualong Tang, Yu Zhang, Vahid Mohmoodian, and Hadi Charkhgard. Automated flight planning for high-density urban air mobility. *Transportation Research C*, 131:103324, 2021.

[78] Laurie A. Garrow, Patricia Mokhtarian, Brian German, and Sreekar-Shashank Boddupalli. *Commuting in the Age of the Jetsons: A Market Segmentation Analysis of Autonomous Ground Vehicles and Air Taxis in Five Large U.S. Cities*. 2020. doi: 10.2514/6.2020-3258. URL https://arc.aiaa.org/doi/abs/10.2514/6.2020-3258.

[79] Conor Hill and Laurie A. Garrow. *A Market Segmentation Analysis for an eVTOL Air Taxi Shuttle*. 2021. doi: 10.2514/6.2021-3183. URL https://arc.aiaa.org/doi/abs/10.2514/6.2021-3183.

[80] Jean-François Cordeau and Gilbert Laporte. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153(1):29–46, September 2007. ISSN 1572-9338. doi: 10.1007/s10479-007-0170-8. URL https://doi.org/10.1007/s10479-007-0170-8.

[81] Stefan Ropke, Jean-François Cordeau, and Gilbert Laporte. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, 49(4):258–272, 2007. ISSN 1097-0037. doi: 10.1002/net.20177. URL

https://onlinelibrary.wiley.com/doi/abs/10.1002/net.20177. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.20177.

[82] Timo Gschwind and Stefan Irnich. Effective Handling of Dynamic Time Windows and Its Application to Solving the Dial-a-Ride Problem. *Transportation Science*, 49(2):335–354, September 2014. ISSN 0041-1655. doi: 10.1287/trsc.2014.0531. URL https://pubsonline.informs.org/doi/10.1287/trsc.2014.0531. Publisher: INFORMS.

[83] Timo Gschwind and Michael Drexl. Adaptive Large Neighborhood Search with a Constant-Time Feasibility Test for the Dial-a-Ride Problem. *Transportation Science*, 53(2):480–491, March 2019. ISSN 0041-1655. doi: 10.1287/trsc.2018.0837. URL http://pubsonline.informs.org/doi/abs/10.1287/trsc.2018.0837. Publisher: INFORMS.

[84] K. N. Genikomsakis and Georgios Mitrentsis. A computationally efficient simulation model for estimating energy consumption of electric vehicles in the context of route planning applications. *Transportation Research Part D: Transport and Environment*, 50:98–118, 2017. doi: 10.1016/J.TRD.2016.10.014.

[85] Dominik Goeke and Michael Schneider. Routing a mixed fleet of electric and conventional vehicles. *European Journal of Operational Research*, 245(1):81–99, August 2015. ISSN 0377-2217. doi: 10.1016/j.ejor.2015.01.049. URL http://www.sciencedirect.com/science/article/pii/S0377221715000697.

[86] Samuel Pelletier, Ola Jabali, and Gilbert Laporte. Charge scheduling for electric freight vehicles. *Transportation Research Part B: Methodological*, 115:246–269, September 2018. ISSN 0191-2615. doi: 10.1016/j.trb.2018.07.010. URL http://www.sciencedirect.com/science/article/pii/S0191261517308871.

[87] Mohamed Amine Masmoudi, Manar Hosny, Emrah Demir, Konstantinos N. Genikomsakis, and Naoufel Cheikhrouhou. The dial-a-ride problem with electric vehicles and battery swapping stations. *Transportation Research Part E: Logistics and Transportation Review*, 118:392–420, October 2018. ISSN 1366-5545. doi: 10.1016/j.tre.2018.08.005. URL http://www.sciencedirect.com/science/article/pii/S1366554517310086.

[88] Claudia Bongiovanni, Mor Kaspi, and Nikolas Geroliminis. The electric autonomous dial-a-ride problem. *Transportation Research Part B: Methodological*, 122:436–456, April 2019. ISSN 0191-2615. doi: 10.1016/j.trb.2019.03.004. URL http://www.sciencedirect.com/science/article/pii/S0191261517309669.

[89] Luke Marshall, Natashia Boland, Martin Savelsbergh, and Mike Hewitt. Interval-based Dynamic Discretization Discovery for Solving the Continuous-Time Service Network Design Problem. page 36, 2019.

[90] Sahar Bsaybes, Alain Quilliot, and Annegret K. Wagler. Fleet management for autonomous vehicles using flows in time-expanded networks. *TOP*, 27(2):288–311, July 2019. ISSN

1863-8279. doi: 10.1007/s11750-019-00506-4. URL https://doi.org/10.1007/s11750-019-00506-4.

[91] Natashia Boland, Mike Hewitt, Luke Marshall, and Martin Savelsbergh. The price of discretizing time: a study in service network design. *EURO Journal on Transportation and Logistics*, 8, March 2018. doi: 10.1007/s13676-018-0119-x.

[92] Stefan Irnich and Guy Desaulniers. Shortest Path Problems with Resource Constraints. In *Column Generation*, pages 33–65. Springer, March 2006. doi: 10.1007/0-387-25486-2_2. Journal Abbreviation: Column Generation.

[93] Nicholas Polaczyk, Enzo Trombino, Dr Peng Wei, and Dr Mihaela Mitici. A Review of Current Technology and Research in Urban On-Demand Air Mobility Applications. In *Proceedings of the Vertical Flight Society Autonomous VTOL Technical Meeting and Electric VTOL Symposium*, page 11, 2019.

[94] Alessandro Bacchini and Enrico Cestino. Electric VTOL Configurations Comparison. *Aerospace*, 6(26), 2019.

[95] Uber. UberAir Vehicle Requirements and Missions, 2018. URL https://s3.amazonaws.com/uber-static/elevate/Summary+Mission+and+Requirements.pdf.

[96] Douglas. Steinley. K-means clustering: A half-century synthesis. *British Journal of Mathematical and Statistical Psychology*, 59(1):1–34, May 2006. ISSN 00071102. doi: 10.1348/000711005X48266. URL http://doi.wiley.com/10.1348/000711005X48266.

[97] THOMAS M. CORSI, CURTIS M. GRIMM, and JANE FEITLER. The Impact of Deregulation on LTL Motor Carriers: Size, Structure, and Organization. *Transportation Journal*, 32(2):24–31, 1992. ISSN 0041-1612. URL http://www.jstor.org/stable/20713155. Publisher: Penn State University Press.

[98] Capgemini Research Institute. The Last-Mile Delivery Challenge. Technical report, Capgemini Research Institute, 2019. URL https://www.capgemini.com/insights/research-library/the-last-mile-delivery-challenge/.

[99] Johnny T Doo, Marilena D Pavel, Arnaud Didey, Craig Hange, Moffett Field, Nathan P Diller, Michael A Tsairides, Michael Smith, Bell Textron, Edward Bennet, Michael Bromfield, Jessie Mooberry, and Airbus Sv. NASA Electric Vertical Takeoff and Landing (eVTOL) Aircraft Technology for Public Services – A White Paper. Technical report, NASA, August 2021.