SERVICE NETWORK DESIGN FOR PARCEL TRUCKING

A Dissertation
Presented to
The Academic Faculty

By

Adolfo Rocco

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial and System Engineering

Georgia Institute of Technology

May 2021

SERVICE NETWORK DESIGN FOR PARCEL TRUCKING

Thesis committee:

Dr. Alan Erera, Advisor
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Alejandro Toriello, Advisor
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Martin Savelsbergh, Co-Advisor
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Benoit Montreuil
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. German Riano
Worldwide Capacity Planning, Operations Research Group
Amazon, U.S.

Date approved: May, 2021

To my late grandmother, my mami

ACKNOWLEDGMENTS

First of all, I would like to express my full gratefulness to my advisors: Prof. Alan Erera, Prof. Alejandro Toriello and Prof. Martin Savelsbergh during my PhD journey. I have learned plenty from them, not only as a source of knowledge but also as human beings.

My sincere thanks to the members of my thesis committee, Prof. Benoit Montreuil and Dr. German Riano. It is a pleasure to have the opportunity to present my thesis to you.

To all the friends I made in Atlanta, for all the crazy, funny and unforgettable moments we lived.

To all my friends and all my family from Chile, who I hold them in my heart.

Specially, I would like to give my full appreciation to my mother, my late grandmother, and my late uncles who have been the support and the fuel of my life to pursue my dreams.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Last-mile logistics is an essential part of the economy and it makes possible the transportation of goods from producers to end-consumers. As a result of the explosive growth of e-commerce in the past decade, demand for delivery of packages directly to consumers is soaring and is expected to grow by 78% globally by 2030. One of the primary objectives of package logistics is to transport shipments as quickly and cost-effectively as possible. In this thesis, we consider a number of important service planning problems for the package trucking carriers that are providing essential service to this e-commerce revolution.

Logistics and service network design models are the important tools for developing novel enabling technology and enhancing current practices in parcel and freight logistics. Service network design models are used to create consolidation plans for carriers, providing the planner with the choice of transfer paths for shipments through a network of consolidation terminals and the services and resources necessary to execute these paths. In the most complex package systems, packages may be consolidated and transferred between vehicles during the last-mile phase of transportation, requiring careful coordination of vehicle schedules. For packages moving between regions, linehaul consolidation plans specify shipment flow paths between origin and destination terminals. Feasible flow plans meet service time requirements, and it can be advantageous if these plans are simple to operate in practice; one example of a simplifying feature for a plan is when flow paths form a *directed in-tree* into each destination terminal such that packages headed to a common destination are transferred to the same next terminal regardless of their origins. The selection of hub terminals to use for package transfer between vehicles in the linehaul network is another important decision in package consolidation network design. In this thesis, we address decision problems in all of the areas discussed above. We develop large-scale package express service network design methods using integer programming optimization models specified on flat network models that capture important timing constraints to ensure that package flows meet service constraints. The first part of the thesis focuses on a detailed intracity scheduling service network design problem for megacities, whereas, the last two parts focus on linehaul consolidation planning.

In Chapter 2, we present a service network design problem in the context of intra-city package courier service in megacities. The problem consists of designing a cost-effective consolidation network to transfer packages from their origins to their destinations within cities while meeting committed service requirements. In this study, we focus on shuttle activities and develop optimization technology for the design of shuttle services using novel rate-based models to determine package flow paths as well as vehicle routes. To ensure operational simplicity, we build repeatable vehicle operating cycles, which are executed throughout the day, that provide adequate capacity for transfers along flow paths and meet package timing requirements. A computational study using data from a large Chinese package company demonstrates that the technology produces a cost-effective service network design for shuttle schedules with excellent on-time performance.

In Chapter 3, we present a strategic hub selection problem within the context of service network design for a package courier system operating fast time-definite services between urban areas. The selection of intermediate hubs for package transfer adds an additional layer of complexity to traditional service network flow planning problems. The aim is to select the hub terminals that enable cost-effective consolidation designs by serving as linehaul transfer locations along package flow paths, while leaving the remaining facilities to serve only as gateways to last-mile operations. We develop a cost-effective greedy heuristic approach that solves tractable integer programming models to add a single intermediate hub to the existing set on each iteration. We develop three such IP-based greedy heuristic variants: (1) Greedy-Hub, (2) Greedy-Hub-Full which solves larger IPs on each iteration, and (3) GRASP-Hub that randomizes the selection of intermediate hubs. Each of these approaches only considers flow paths with at most one intermediate hub transfer. A computational study shows that the greedy approach selects geographically-distributed and cost-effective hubs for package transfer, and moreover, the heuristic outperforms the full optimization model by a 20% gap difference for the relevant test instances.

Finally, in Chapter 4, we develop a new approach for solving the flow planning problem of service network design for large-scale networks with timing constraints; in this chapter, the selection of intermediate transfer hubs is fixed in advance. We seek plans that can be executed in practice, and thus focus on a new generalization of the concept of an in-tree flow plan. We introduce a so-called generalized in-tree, referred to as GIT, which has useful operational benefits and specifies a common next terminal in the flow path for all commodities arriving to a transfer terminal with the same destination and similar remaining time to make service. We develop three approaches for finding flow plans conforming to a GIT structure: (1) a GIT-based approach, (2) a path-based approach and (3) an optimized GIT-based formulation. Both the GIT-based approach and the path-based approach use a novel dynamic variable generation scheme, similar to column generation for linear programming, for introducing new candidate GITs or paths into a master integer programming formulation. We demonstrate, via a computational study, the quality of the solutions found by these various approaches. First, we demonstrate that building flow plans with paths that may contain more than one intermediate stop can improve the flow plan costs for the same instances as in Chapter 3 by 3% when the same intermediate hub terminals are selected. Then, we show ,using a variety of test instances, that imposing a discretized GIT structure that groups remaining times into fixed-width buckets of 2 hours or 4 hours leads to solutions that are only 2% to 4% more costly than those that do not require GIT structure but significantly simpler to operationalize.

CHAPTER 1
INTRODUCTION AND BACKGROUND

Middle and last-mile logistics have become an essential part of the economy, and improvements to these systems have been a key enabler of modern supply systems that move goods from producers to end-consumers. The last leg of the logistics chain, so-called last-mile logistics (also known as city logistics), refers to the very last step of the logistic delivery process when a parcel is moved from a transportation hub or fulfillment center to its final destination. Last-mile logistics is used to move goods to retail stores for purchase or directly to consumers at their homes or work locations. Last-mile logistics is perhaps the most important and difficult-to-plan component of the supply chain because it is often the most expensive phase of freight transportation (accounting often for more than 50% of the total shipping cost), but also because customer service constraints place strict timing constraints on this phase.

Due to the explosive growth of e-commerce in the past decade, demand for last-mile delivery is soaring and is expected to grow by 78% globally by 2030 ([1]); between 2014 to 2019, e-commerce sales ratios nearly tripled globally and recent growth trends have further exploded in 2020 during the COVID-19 global pandemic. To enable this rapid growth in e-commerce, more capacity is required for delivering packages directly to consumers while meeting requirements to do so both quickly and cost-effectively.

In first-mile and middle-mile transportation, packages are often shipped from large-volume origins like fulfillment centers and move between hub terminals in full or nearly-full trailerloads. This economical transportation of packages continues until the last-mile, when packages must be distributed to a large number of individual destinations that each receive only a small shipment. Improving the operational efficiency of last-mile logistics can allow carriers to maintain (or outsource) a smaller fleet of delivery vehicles and drivers to ship the same number of products, and building last-mile systems that utilize consolidation to increase efficiency closer to the delivery points while still meeting customer service expectation is a critical goal.

In general terms, there are two main types of freight shipments: parcel delivery and freight delivery. Whereas the former is concerned with delivering parcels and small packages, the latter involves heavier and large items, usually greater than 30kg, such as furniture and appliances, large boxes, and palletized deliveries to stores. Furthermore, parcel and freight transportation services can be segmented in terms of time allowed from order to delivery. For example, the parcel segment can be divided into deferred, time-definite, same-day and instant delivery while freight shipments can be divided into standard, expedited, or time-definite.

In this thesis, we will consider time-definite delivery of parcels and packages. In terms of global numbers, more than 100 billion parcels were shipped around the globe in 2019 (according to the Pitney Bowes Parcel Shipping Index, [2]). Due to increased uncertainty stimulated by the global pandemic, the report forecasts that volumes could range from as low as 200 billion to as high as 316 billion parcels in six years. China remains the largest international market for package transportation with 63.5 billion parcels shipped in 2019, followed by the US with 14.7 billion parcels shipped in 2019. These enormous demands for package transportation and the high rates of growth create major challenges for large carriers such as DHL Global, China Post, SFExpress and TNT in China, and USPS, UPS and FedEx in the US.

Logistics and service network design models are frequently used tools for developing new enabling technology and improving current practices in parcel and freight logistics. The general goal of all service network design models is to find efficient and effective ways of transporting goods while minimizing either the resources utilized or transportation costs along with imposing relevant tactical and operational restrictions.

In the operations research literature, the standard service network design problem ([3],

[4]) refers to a problem similar to the network design problem of integer programming. The standard service network design problem specifies demands as required freight volumes (or shipments) for a set of origin-destination pairs, where each origin and destination is a node in a network. Arcs represent transportation *services* connecting these nodes and potentially other hub nodes that enable the transfer of freight from one service to another. Service network design then is to select (or install capacity on) a set of services such that freight demands can be served, where each transportation service has costs when used. Depending on the type of service network design formulated, these models will specify which services to operate and with what capacity, how to schedule the resources providing these services, and then how to assign freight shipments to service paths through the designed network.

An extension to standard service network design is to consider the *roles* of terminals as decisions. Standard problems specify as inputs which terminals may be used as transfer hubs and which others are only freight origins and/or destinations, but an important extension is to allow the optimization model to choose which terminals can be used as transfer hubs. Not all terminals may be suitable to perform cross-docking or package sorting activities, for example if buildings are too small and cannot be expanded with appropriate numbers of docks or sorting areas and equipment. Carriers may only wish to configure a small number of buildings to serve as transfer hubs, so then the question arises where to locate such buildings to maximize the cost-effectivenss of consolidation. Some literature addresses hub location and hub role, but these works do not embed these decisions in detailed service network design models. The review in [4] reveals that minimal attention has been given to the hub network design problem.

General flow planning problems in service network design are NP-hard optimization problems and are often formulated as mixed integer programs. The simplest arc cost models are either fixed-charge or fixed-plus-linear for positive arc flows, and more complex models where multiple vehicles with different capacities can be installed on arcs can lead to problems with complicated discontinuous piecewise-linear cost functions in arc flow.

Such problems become very difficult to solve as the number of commodities and arcs increases, and computational challenges are exacerbated when operational constraints like in-tree plans are enforced. To ensure that commodities are transferred from origin to destination while meeting customer service constraints, *path-based formulations* are used which enumerate service-feasible paths in advance. However, such approaches may only be computationally practical when the numbers of commodities and arcs are small and the number of arcs allowed in any flow path is limited to control the number of paths enumerated. On the other hand, allowing flow paths with more transfers can reduce costs significantly when such paths remain time-feasible. Solution approaches that generate paths as needed, such as branch-and-price and heuristic column generation methods, may help improve the search for high-quality solutions to problems of larger scale. Relevant and related works using these techniques are described in [5], [6], [7] and [8].

In this thesis, we build a number of approaches to address the challenges discussed above. We develop computational approaches for solving large-scale package express service network design, and in each approach we rely on the use of flat network models that capture relevant timing constraints without resorting to time-expanded networks that may explode instance sizes. The first part focuses on a detailed intracity consolidation planning and scheduling service network design problem for urban areas, whereas the second and third parts focus on linehaul consolidation planning. In Chapter 2, we develop an optimization technology for the design of last-mile shuttle services using novel *rate-based* models to determine package consolidation paths as well as vehicle routes. In Chapter 3, we propose a tactical hub selection approach which aims to select the hubs (either *gateway hubs* or *local hubs*) that are the most cost-effective to serve as freight transfer hubs. Finally, in Chapter 4, we build operationally-feasible consolidation flow plans given a set of selected hubs in which solutions conform to a generalized in-tree structure that enhances operational realism. Next, we provide some background information and outline our contributions to the topics covered in the following chapters.

1.1 Operations Design for High-velocity Intra-city Package Service

Driven by the growth in e-commerce, demand for high-velocity services is growing. High-velocity services include standard next-day services where packages collected today are delivered tomorrow and also same-day service where pickup and delivery occur on the same day. In this work, we collaborate closely with one of the largest package couriers in China; their business model includes a plan to grow high-velocity services within Chinese megacities with a new operating model.

A consolidation transportation system typically employs a complex network of transfer hubs, where vehicles transport packages between hubs, and packages are unloaded, sorted, and loaded. Routing packages through intermediate hubs is key to achieve the cost savings of freight consolidation, but requires additional time and package handling. Service network design for such systems includes flow and load planning, and determining vehicle route to execute these plans. When timing constraints are critical, [4] notes that the standard optimization modeling approach is to build deterministic time space networks like those originally presented in [9]. In practice, however, these time-expanded networks can be gigantic in size leading to intractable optimization problems. To have hope in solving these problems, assumptions are often made to reduce the size of these networks or even to avoid time space representations (see, for example, the composite formulations in [10]).

In Chapter 2, we present a service network design problem in the context of intra-city courier service in megacities. The considered setting is inspired by the activities of one of the largest courier package companies in China. The problem consists of designing a cost-effective service network to move packages from their origins to their destinations, taking into account committed service levels. The operational network includes two layers; the first layer consists of the network of riders, transporting packages from (to) customer locations to (from) a series of hubs, while the second layer consists of a fleet of shuttles responsible for the inter-hub transfers. In this study, we focus on shuttle activities and develop optimization technology for the design of shuttle services using novel *rate-based* models to determine package as well as vehicle routes. To ensure operational simplicity, we build repeatable vehicle routes taking the form of cycles that are executed throughout the day, considering capacity and timing requirements. A computational study using data from a large Chinese package company shows that our technology produces a cost-effective service network design for shuttle schedules with excellent on-time performance.

1.2 Service Network Design with Hub Selection

Demand for time-definite package services is growing largely due to growth in e-commerce. Such services include same-day, next-day and two-day services, where packages are collected, routed through a consolidation network of terminals, and then delivered to customers with high velocity. Consolidation terminals include *satellites* or *end-of-line* facilities, and *hub* facilities. Satellites serve only as the interface between the local pickup-and-delivery (*last-mile*) subsystem and the linehaul (*middle-mile*) subsystem, while hub terminals also provide intermediate consolidation and transfer locations within the linehaul subsystem. In this work, we collaborate closely with one of the largest package couriers in China; their business model includes a plan to grow high-velocity services between Chinese cities with a new operating model.

In Chapter 3, we present a strategic hub selection problem within the context of service network design for a package courier system operating fast time-definite services between urban areas. In this work, we add another layer of complexity to the service network design problem: the selection of intermediate hubs for freight transfer. In these problems, the aim is to identify which terminals can serve as the most cost-effective intermediate hubs to build time-feasible cross-docking flow paths for all commodities. We develop a cost-effective greedy heuristic approach that adds one intermediate transfer hub to a plan during each iteration. We develop three IP-based heuristic greedy variants: (1) a Greedy-Hub, (2) a Greedy-Hub-Full which solves larger IPs on each iteration, and (3) a GRASP-Hub that randomizes the selection of intermediate hubs. Computational experiments show that the greedy approach selects geographically distributed cost-effective hubs for freight transfer, and in terms of gap performance, the heuristic outperforms the full optimization model by a 20% gap difference for the interesting cases.

1.3 Path Generation Based Heuristic for Service Network Design

In the linehaul planning problem setting of Chapter 3, after hubs are identified for the service network, it is necessary to create an operationally-feasible consolidation flow plan that leads to the lowest operational cost. Such flow plans should not constrain the number of transfer stops for each commodity as long as the flow paths remain time feasible. Furthermore, consolidation plans should be implementable in practice.

In order to enhance operational realism while building flow plans with the lowest possible cost, we develop the notion of a consolidation plan that conforms to a *generalized in-tree* structure for each destination. A generalized in-tree plan is one where commodities that are dispatched outbound from a terminal travel next to the same terminal if they share the same final destination and a similar remaining time to meet service. We develop flow planning models that build plans that conform to generalized in-trees, and then we seek low-cost solutions to these models by creating approaches that create new paths for commodities dynamically during the solution approach. The path generation approach is inspired by the concepts of column generation in which we employ dual values from a useful linear program to compute the reduced costs of arcs. These arc reduced costs are then used in a column pricing scheme which seeks to identify new possible flow paths that should be used in the mixed-integer programming model.

In Chapter 4, we describe several dynamic path generation heuristic approaches that seek to find a set of flow paths conforming to a generalized in-tree (GIT) structure. Each approach relies on solving an approximate variable pricing problem using constrained minimum cost paths with a set of arc reduced costs that are estimated from a related linear program. Generated paths are then used to solve a mixed integer programming model for flow planning, and the process is iterated. We develop three approaches: (1) a relaxation of the problem using a path-based formulation, (2) a GIT-based approach, and (3) an optimized GIT-based formulation. We demonstrate, via a computational study, that the approach can yield high-quality flow plans with 3% lower operational costs than those that can be achieved when a smaller number of paths with fewer transfer options are enumerated in advance, as in Chapter 3. Furthermore, the path-based relaxation produces solutions that conform to a 4-hour discretized GIT structure for all tested small instances, and that closely approximate a GIT structure at the 2-hour and 4-hour discretization for medium and large instances. We also show that imposing more restrictive discretized GIT structures on a problem creates a cost penalty typically between 2% and 4%. All three approaches produce results of similar quality when solving small size instances. However, the path-based and the optimized GIT-based approaches strongly outperform the GIT-based approach by more than 10% of gap performance for the large size instances.

CHAPTER 2
OPERATIONS DESIGN FOR HIGH-VELOCITY INTRA-CITY PACKAGE SERVICE

2.1 Introduction

We consider an operations network design problem for a package courier operating high-velocity services within large urban areas. In this setting, high-velocity services include standard next-day services where packages collected today are delivered tomorrow and also same-day service where pickup and delivery occur on the same day. Demand for high-velocity services is growing. In this work, we collaborate closely with one of the largest package couriers in China; their business model includes a plan to grow high-velocity services within Chinese megacities with a new operating model. We build novel optimization technology to configure vehicle operations using new and novel *rate-based* routing and network design models that use parcel demand rates per time as inputs, and that determine both route capacity and service-level feasibility with vehicle flow rates per time between locations induced by repeated execution of vehicle routes during an operating day.

Consider a system with a number of small hub terminals throughout an urban area. These locations, denoted *local hubs* (LH), are used for consolidation of packages into and out of a set of small geographic service regions into which the urban area has been partitioned. Due to the congested urban environment, the *couriers* who pick up and deliver goods directly from and to customers do not operate large vehicles; instead, they walk or use small delivery bikes with limited package capacity. Many couriers operate within each local hub service region, but they do not visit the local hub and instead spend the day working within their assigned *unit zone*. Packages are transferred to and from couriers within their unit zones via a fleet of small-capacity transfer vehicles known as *riders*. Riders transfer packages with a courier at a designated *access hub* (AH) location either synchronously via timed meet-ups, or asynchronously through the use of parcel lockers.

Packages are transported between service regions of different local hubs via a second set of vehicles, known as *shuttles*. Shuttle vehicles are larger than rider vehicles, since they only need to stop at local hub locations. In a large urban area characterized by many service regions and local hubs, it is likely not economical to schedule direct shuttle movements between all pairs of local hubs. Packages can be cross-docked between shuttles at local hubs to enable non-direct service. Overnight storage of parcels is not allowed at local hubs, and instead only at a small set of larger facilities denoted *gateway hubs* (GH) that also provide intercity service for packages moving into or out of the urban area. Shuttle services thus also transfer outbound intercity packages from LHs to GHs, and inbound packages from GHs to LHs.

In this research, we consider approaches to design shuttle vehicle operations, and associated cross-dock transfers, to enable effective intra-city transfer of packages. The objective is to create a design that moves packages between LHs and GHs to meet timing requirements, while minimizing the cost of providing the services. Unlike traditional approaches for city logistics design, we seek to construct repeatable service cycles for shuttles that can be executed during (a portion of) the operating day to provide continuous transfer service.

One of the first models for service network design was developed by [11]. The authors consider the context of express shipment service in which shipments must be picked up and delivered within specified time intervals (e.g., 24 hours, 48 hours or 3–5 days) and develop a column generation approach for the design of schedules. Subsequent studies by [10] and [12] focus on the next-day air operations developing a technology that determines aircraft routes, routing of packages and fleet assignments. In these studies, the authors developed what they call the composite variable formulation, which is an extended formulation that makes use of aggregated variables, improving lower bounds of the linear relaxation compared to conventional approaches. [13] developed a modeling framework with an iterative approach, which is based on the idea of reducing the size of the problem in every iteration. Our shuttle scheduling and routing has similarities to the context of less than truckload (LTL) on long-haul transportation. LTL carriers are one of the main building blocks of order fulfillment in both online and brick-and-mortar stores. In order to deliver goods on time and in a cost-effective manner, LTL carriers consolidate shipments via planning and coordinating the paths of different shipments in both space and time. The service network design problem ([3],[4]) is the common tool used in these planning processes, by providing the planner with the choice of paths for shipments and the services or resources necessary to execute them. Building such a plan involves selecting the services to operate, their schedules, and then executing them by routing shipments through the selected service network. In that sense, the problem of shuttle scheduling can be seen as an intra-city variant of an LTL carrier problem. [14] consider the LTL problem for motor carriers determining how freight should be routed through the network while balancing level of service and costs. They develop a local improvement heuristic given the large size of the problem they address. An extension of this work, in [15], presents an enhanced local improvement heuristic that adds and drops direct services, rerouting the flow to determine the effect of the switch on costs. In a related subsequent work, [16], aim to synthesize the algorithms and their implementations and show the relevant interaction among algorithmic approach, software architecture and implementation.

A variant of the service network design, referred to as the frequency service network design problem aims at determining the type and frequency of services on freight routes. In such a problem, demand is usually described as flows between the various origin/destination pairs. Such a problem is referred to as multi-commodity network design, in which commodities are recognized based on their weight, volume, origin, destination, departure time and delivery condition ([17], [3]), however, it is more and more difficult to identify service preferences of shippers by the appearance of the goods or the commodity type ([18]).

In a recent work, [19] present a frequency-based model for freight service network design aiming to improve service performance. [20] consider a setting where resources at terminals are limited; a service network model for freight consolidation carriers selecting the services and routes for commodities. [21] focus on the tactical planning process for city logistics, that the authors identify as the day-before problem, in the general case of two-tiered systems, while [22] present an application of the methodology proposed for the single-tiered case. In a similar work, [23] address a tactical plan of a two-tier city logistic system considering uncertainty in the forecast demand, which is an aspect not commonly taken into account explicitly in tactical planning. [24] also consider uncertainties in demand concluding that consolidation and hub-and-spoke systems deliver better solutions when there is stochastic demand due to the fact that the solution provides a sort of hedging. In a similar work, [25], focus on addressing the services of the first tier system of a two-tier structure.

Excellent reviews of literature on service network design are provided in [26] and related to freight transportation in [3], which focus on modeling and mathematical programming approaches. [4] present a more recent review of different problem formulations and a comparison of solution frameworks for service network design for freight transportation problems.

A branch-and-price approach is developed in [6] to solve service network design problem with asset management, which is initially presented in [27], addressing the vehicle management decisions and service network design aspects jointly. [28] present an integer formulation for load planning in the context of LTL freight transportation. A similar work for LTL in the context of large-scale problems is presented in [29]. A math-heuristic for service network design with design-balanced requirements that uses cutting plane methods to obtain tighter bounds and fixing variables to reduce problem size is presented in [30].

[31] discusses the challenges and opportunities in city logistics, in which it remarks on the importance of developing highly dynamic models that could be able to handle real-time information. In a more recent paper, [32] present a review on cluster classification prob-

lems, mathematical models and their solution methods for urban transportation network design problems and also propose future research directions. A similar but older review of urban transportation network design problems is presented in [33].

To the best of our knowledge, no prior work addresses the challenges related to the service network design problem associated with high-velocity intra-city courier services. In summary, our contributions are:

- We propose a data-driven approach to design the network and to schedule the movements within the network such that the offered services between different origin-destination pairs within the city become operationally feasible and efficient.

- We propose a rate-based model; discrete demands are converted into demand rates per hour. The proposed model takes demand rates as input and constructs shuttle cycles guaranteeing the capacity and dispatch frequencies required to meet the desired service level. This approach considers a flat network incorporating the frequency of dispatch frequency on each arc needed to ensure on-time deliveries.

- To ensure time feasibility of the obtained solution, we conduct a post-processing phase. First, the generated routes are evaluated on a time-space network and if necessary, simple cycles are added to the solution. Then, using a cost-benefit study, the least beneficial cycles are removed, while preserving a target minimum service guarantee.

- A comprehensive case study is conducted based on the historical data received from one of the leading Chinese courier service providers. The outcome of sensitivity analysis demonstrates the robustness of our proposed routes even in scenarios with significant shifts in demand patterns.

The remainder of the chapter is structured as follows. In Section 4.3, we present the setting and description of the problem. Next, Section 2.3 describes the mathematical formulations of our approach. In Section 2.4, we present the post-processing phase of shuttle scheduling. Computational study is reported in Section 2.5. Finally, we present conclusions and future research directions in Section 2.6.

2.2 Problem Setting

We consider an intra-city service network defined on a multigraph $\mathcal{G} = (\mathcal{U}, \mathcal{E})$ with node set $\mathcal{U}$ representing the set of terminals and arc set $\mathcal{E}$ representing (directed) transportation connections between terminals. The terminal set $\mathcal{U}$ comprises three categories of hubs: gateway hubs (GH), local hubs (LH), and access hubs (AH), represented by $\mathcal{U}^G$, $\mathcal{U}^L$, and $\mathcal{U}^A$, respectively. Most of the sorting activities occur at GHs. The LHs have limited capacity for sorting activities and are mainly used for transshipment purposes. The AHs act as local pick-up/drop-off locations. Many-to-one AH-LH and LH-GH assignments are considered. That is, for each customer location, one can consider an origin AH, an origin LH, an origin GH, a destination GH, a destination LH, and finally a destination AH. While this pre-assignment is fixed, packages may skip visits to GHs depending on their service classes.

Note that, while the network is defined as an intra-city service network, the set of shipments moved may be a mix of intra- and inter-city shipments. We consider a heterogeneous fleet of vehicles $\mathcal{V}$, with N^v and Q^v being the number of available vehicle type v and the capacity of each vehicle of type v, respectively. The cost of traversing an arc $e \in \mathcal{E}$ by a vehicle of type v is denoted by c_e^v and the time of traversing the arc is denoted by t_e^v (in a general setting, different vehicle types may have different speeds). In the multigraph $\mathcal{G}$, each node (terminal) pair is connected with potentially multiple arcs, representing different vehicle types.

Let $\mathcal{S}$ be the set of intra-city service classes offered. The set includes same-day (SD) and next-day (ND). The inter-city packages are sorted at GHs and leave the origin city overnight. Each service class is defined by two cut-off times: (1) the latest time a package may be picked up from its origin to be eligible for a given service class, and (2) the latest time a package can be delivered at its destination based on the committed service class. A package may go through different hubs and may be loaded on and unloaded from multiple vehicles on the path from its origin to its destination. The path a package takes and the set of hubs it visits are impacted by the package service class. A package visit to a hub may involve sorting operations or cross docking. As a general rule, all packages with service class ND should visit at least one GH for sorting activities overnight. Conversely, same-day packages do not typically visit a GH for a sorting purpose (they may still be cross-docked at a GH).

2.2.1 Daily routines

Three types of transportation activity take place from the moment a package is picked up from its origin to the time it is delivered to its destination. Following a service request by a customer, the package is first picked up by a *courier* from the customer's location at a specified time. A courier may pick up several packages from different customers within the same region, referred to as a *unit zone* assigned to the courier. Packages picked up by a courier are then dropped off at an AH. A fleet of *riders* collect packages from the AHs and deliver them to LHs. A rider tour may include visits to several AHs. Outbound inter-city packages as well as intra-city packages with ND service class are consolidated at LHs and are sent to GHs. After being sorted at the GHs, inbound intra-city packages are transferred to their destination LHs. Inter-city packages may visit one or more GHs before reaching their destination LHs (whatever happens beyond the borders of the intra-city network is out of the scope of this chapter). Same-day intra-city packages are sorted at their origin LHs and are then sent to their destination LHs. The movements between LHs and GHs are operated by *shuttles*. Packages at the destination LHs are again picked up by the riders who are responsible for transferring them to their destination AHs. Couriers collect packages from the destination AHs and deliver them to their ultimate destinations. An illustration of the intra-city network and the interaction among couriers, riders and shuttles is shown in

Figure 2.1.

Notice that courier, rider, and shuttle routes may potentially incorporate pickups and deliveries on the same routes, as multiple locations may be visited. However, due to the pre-assignments of AHs to LHs, a rider can only visit AHs assigned to a single LH. As for the shuttles, although a shuttle may visit LHs assigned to different GHs, the loading and unloading of packages on the vehicles are done such that the package paths respect the hierarchical structure of the network.

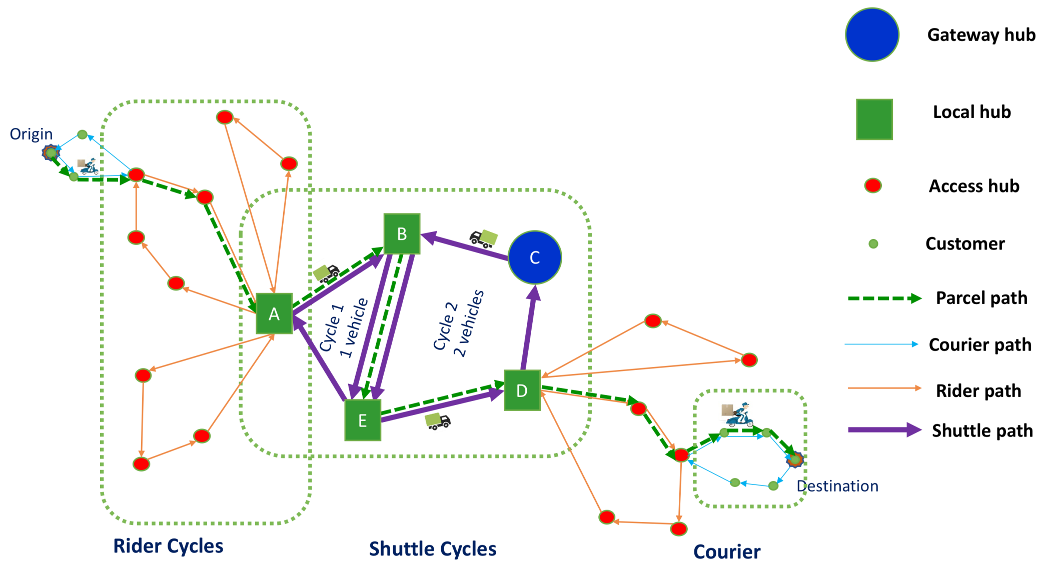

Figure 2.1: Intra-city Network

Based on the above descriptions, one can notice that apart from same-day packages, all other packages visit at least one GH on their path from their origin to their destination. Since providing same-day service may involve certain requirements, the shuttle activities throughout a day are categorized into two regimes: Morning (Regime 1 - R1) and Afternoon (Regime 2 - R2). These regimes are differentiated based on a cutoff time. Each regime involves packages arriving in the system during its corresponding part of the day. Regimes are also defined based on the overall demand patterns. While R1 mostly incorporates the transfer of ND intra-city and inbound inter-city packages from GHs to their destination LHs and subsequently AHs, R2 involves the transfer of ND intra-city and outbound inter-city packages picked up during the day to the GHs. Same-day intra-city demand is almost uniformly distributed over the two regimes. It is worth mentioning that at a micro level, fluctuations in demand patterns throughout a day are not significant enough to justify major changes at the operational level. As a result, rider schedules do not undergo changes during different periods of the day, i.e., R1 and R2. We now describe the two regimes in more detail.

Regime 1 (R1): Morning This regime involves packages entering the system during the period starting from 8:00 am to 12:00 pm. The main activities during this regime include transferring ND intra-city packages picked up the day before in addition to inbound inter-city packages sorted overnight at a GH, to LHs. These packages are then transferred to AHs by riders and from there, they are delivered to their final destinations by the couriers. This regime also serves same-day packages picked up in the morning and midday. Packages picked up by the couriers before the same-day service cut-off time are guaranteed same-day delivery; they are delivered by the same-day delivery cut-off time. Intra-city packages arriving at their origin LHs are sorted at the LHs and are transferred to their destination LHs through the network of shuttles (without necessarily visiting any GH in between).

Regime 2 (R2): Afternoon This regime corresponds to packages entering the system during the period starting from 12:00 pm to 4:00 pm. During this regime, outbound inter-city and intra-city packages with service classes ND are transferred to their origin GHs by a fleet of shuttles. These packages are sorted at the GHs and are potentially dispatched to their destination GHs overnight. The next morning, the packages are transferred to their destination LHs during the morning regime. Similarly, this regime serves the afternoon part of same-day delivery packages. Intra-city packages picked up by the couriers after same-day service cut-off time are then transferred to the assigned GH, to be delivered next day morning in R1.

The main focus of this chapter is on shuttle operations. Since the demand arrives over time, one way to guarantee high quality of service is to generate a set of shuttle cycles that repeat throughout the day or at least part of the day. A shuttle cycle consists of a sequence of visits to a subset of LHs and/or GHs. Additionally, a cycle is characterized by its vehicle type, start location and time. The cycles should be designed to ensure the existence of at least one time feasible path between each origin-destination pair and a service class, if the service is offered. Moreover, the cycles must provide a certain frequency of visits to terminals to maintain the required service level, i.e., high on-time delivery rate. Since demand patterns of different regimes are drastically different, it seems reasonable to potentially generate different paths and cycles for different regimes. Based on this idea, we aim at generating baseline static shuttle cycles that ensure a reasonable service quality while maintaining operational costs as efficient as possible.

2.2.2 Shuttle Cycle Operations Periods

As stated before, the demand pattern and consequently the set of shuttle cycles in the two regimes may be quite different. That is, demand for a given origin-destination pair could be nonzero in one regime and zero in the other regime. As a result, to ensure that packages in each regime are delivered to their destination, shuttle cycle operation periods associated with the regimes are extended beyond the normal hours of the regimes. Specifically, while the regimes are switched at 12:00 pm, cycle operations of R1 may continue until 4:00 pm, and similarly, cycles from R2 continue operating until 8:00 pm. Since cycle operations of R2 begin at 12:00 pm, the period of 12:00 pm to 4:00 pm is covered by both regimes. In this overlapping period, commodities are permitted to use resources from either regimes. Cycles associated with each regime start their operations at the beginning of the regime.

We consider deterministic travel times, transferring times and waiting times. Hence, given a start location for a cycle, we can determine where and when the cycle will finish operating at the end of the corresponding regime.

2.3 Shuttle Scheduling

In this section, we only focus on the subnetwork $\mathcal{G}^S \subset \mathcal{G}$ that concerns shuttle routes, namely the set of terminals $\mathcal{U}^S = \mathcal{U}^G \cup \mathcal{U}^L$. In such a network, package batches arrive at origin LHs in waves by riders. Each package batch delivered by a rider may contain inter-city packages or intra-city packages with service class ND that must be transferred to the GHs, or same-day packages that need to be transferred to their destination LHs.

In this network, we define a commodity k to be the set of packages sharing the same origin, destination, origin regime, and service class. Let $\mathcal{K}$ be the set of all commodities in the system, where a commodity $k \in \mathcal{K}$ is identified by a tuple $(o_k, d_k, \tau_k, s_k, q_k)$, where $o_k, d_k \in \mathcal{U}^S$ are the origin and destination respectively, $\tau_k \in \{R1, R2\}$ is the origin regime, $s_k \in \mathcal{S}$ is the service class and finally q_k is the commodity size measured in packages per hour. In practice, commodities arrive in LHs and GHs in waves or batches based on discrete arrivals of the riders and shuttles. In order to capture timing in the model, we assume the demand is uniformly distributed throughout a regime. That is, given a regime of H hours, the arrival rate of commodity k in that regime is assumed to be $q_k = \sum_{h \in H} q_k^h / H$, with q_k^h being the number of packages of commodity k arriving in hour h. Note too that the service class corresponds to the total time available between the pickup and delivery times from the origin to the destination of packages. Thus, only part of this time can be used for the shuttle operations. Let δ_k be the available time between the origin and destination of commodity k for shuttle operations.

The main objective is to move commodities from their origins to their destinations taking into account their due dates. To do so, for each commodity, we choose a path from its origin to its destination. A path consists of a direct arc from the origin to the destination of the commodity, or a sequence of arcs, with possible transfers at intermediate terminals. We consider a transfer time t_u at each terminal $u \in \mathcal{U}^S$, which represents the handling time required to transfer a commodity through the terminal. Handling time t_u may be considered independent of commodity flow (fixed handling time) or as a function of the flow (variable handling time).

Given the fixed fleet size, commodity paths are designed to maximize consolidation opportunities, which results in minimizing the operational costs (traveling cost and handling costs). We aim to determine vehicle routes such that all paths are covered with the frequency needed to respect service classes. Vehicle routes take the form of cycles, where a cycle is a circuit that starts and ends at the same terminal $u \in \mathcal{U}^S$. Obviously, the number of cycles performed by each vehicle type v may not exceed the available fleet, N^v. We assume each cycle is performed continuously throughout the corresponding regime.

Due to significant differences in demand patterns, we address each regime separately. For each regime, shuttle schedules are constructed through a three-phase procedure. For each regime, we first identify one path per commodity. Then, we generate cycles that cover the paths created in the first phase. In the third phase, taking the shuttle cycles generated in phase two, we potentially re-path the commodities. Each of these phases are described in detail in what follows.

2.3.1 Phase 1: Commodity Path Selection

The main goal of this phase is to generate time-feasible paths for each commodity $k \in \mathcal{K}$, maximizing consolidation opportunities. We define a path p as a sequence of terminals (or a sequence of arcs) $(u_1 = o_k, u_2, ..., u_k = d_k)$ that commodity k takes from its origin to its destination, where each (u_i, u_j) is an arc in $\mathcal{G}^S$. We restrict the maximum number of transfers or intermediate terminals along a path to T_k^P. The parameter T_k^P may be the same for all commodities or it may depend on factors such as arrival rate (e.g., the higher the arrival rate, the smaller is T_k^P). The shape of a commodity path is influenced by the commodity's arrival rate and its service class.

A potential commodity path is said to be time-feasible if the sum of the travel times of all arcs of the path, the handling times at all its intermediate terminals, and expected waiting times at the terminals along the paths is no greater than the available time, δ_k. That is,

$$\sum_{e \in p} t_e + \sum_{u \in p} t_u + \sum_{u \in p \setminus d_k} E(w_u^p) \leq \delta_k \tag{2.1}$$

where $E(w_u^p)$ is the expected waiting time along path p at terminal u. Let $W^p = \sum_{u \in p \setminus d_k} E(w_u^p)$ be the total expected waiting time along path p. Expected waiting time at terminal u along path p depends on the outbound dispatch frequencies along p at u (frequency of dispatch frequency on the arc leaving u along p).

Dispatch frequencies on arcs are driven both by the flow requirements and the maximum allowable waiting time. We define the maximum allowable waiting time of commodity k along path p as

$$\hat{w}_k^p = \delta_k - \sum_{e \in p} t_e - \sum_{u \in p \setminus d_k} t_u. \tag{2.2}$$

where for simplicity of notation, t_e represents the travel time w.r.t. the vehicle type along e.

To select a commodity path, we first identify a set of candidate time-feasible paths, $\mathcal{P}_k$ for each commodity $k \in \mathcal{K}$. We restrict the set of time-feasible paths considered for each commodity $k \in \mathcal{K}$ as follows:

- Consider at most T_k^P intermediate terminals for the path.
- Consider intermediate terminals such that the total travel time and handling time of the path is at most $1 + \beta$ times the travel time of the direct path. (The paths should have enough buffer to absorb potential waiting time at their intermediate terminals, but still remain time feasible.)

In addition to the parameters introduced before, let binary variable x_k^p equal 1 if path p is selected for commodity k, and 0 otherwise. Also, let z_e^v be a variable representing the dispatch frequency of truck type v on arc e. The binary parameter γ_e^p equals 1 if arc e is part of path p. The problem of choosing one path per commodity such that the (variable) cost of the arcs is minimized while maximizing the opportunity for consolidation in the network takes the following form.

$$\begin{aligned}
\min \quad & \sum_{v\in\mathcal{V}}\sum_{e\in\mathcal{E}} c_e^v z_e^v & & (2.3)\\
\text{s.t} \quad & \sum_{p\in P_k} x_k^p = 1 & \forall k \in \mathcal{K} & \quad (2.4)\\
& \sum_{k\in\mathcal{K}}\sum_{p\in P_k} \gamma_e^p q_k x_k^p \le \sum_{v\in\mathcal{V}} z_e^v Q^v & \forall e \in \mathcal{E} & \quad (2.5)\\
& |p| \frac{x_k^p}{2\hat{w}_k^p} \le \sum_{r\in\mathcal{R}} z_e^v & \forall e \in p, \forall p \in P_k, \forall k \in \mathcal{K} & \quad (2.6)\\
& x_k^p \in \{0,1\} & \forall p \in P_k, \forall k \in \mathcal{K} & \quad (2.7)\\
& z_e^v \ge 0 & \forall e \in \mathcal{E}, \forall v \in \mathcal{V} & \quad (2.8)
\end{aligned}$$

The objective function (2.3) minimizes the total variable costs incurred along the chosen arcs. Constraints (2.4) state that exactly one path per commodity is selected. Constraints (2.5) guarantee that the dispatch frequency on each arc is enough to handle the flows of all commodity paths using that arc. Constraints (2.6) ensure that the dispatch frequency on each arc is enough so that the maximum allowable waiting time along the path is not violated (distributing the maximum allowable waiting time along the path uniformly between terminals of the path). Finally, constraints (2.7) and (2.8) indicate the type and domain of the variables.

2.3.2 Phase 2: Cycle Selection

The outcome of model (2.3)-(2.8) is the set of arcs, $\hat{\mathcal{E}}$, requiring non-zero minimum dispatch frequency z_e^v. The vector of $z_e^v \quad \forall e \in \mathcal{E}, v \in \mathcal{V}$, obtained from the path model is considered as the main input to the cycle selection phase.

From a practical point of view, it may be more difficult to synchronize or implement long cycles, either in terms of the number of arcs or the length of the cycle. The two restrictions together aim at keeping the drivers in limited distances from their home terminals (the terminal where they start their shift). This allows us to possibly enumerate all possible cycles, when the set $\mathcal{U}^S$ is not prohibitively large. The cycle selection phase is based on the two following steps:

1. We create a set of feasible cycles of vehicle type $v, \mathcal{C}^v$, based on the following criteria:

    - Consider all possible cycles with at most T^{SC} arcs.
    - Consider cycles no longer than L^{SC} total unit of travel and handling time.

2. We solve the model (2.9)-(2.12) to choose the best set of cycles that guarantees the required minimum dispatch frequency of the arcs of the network, obtained from the path model.

Let integer variable y_c denote the number of times that cycle $c \in \bigcup_{v\in\mathcal{V}} \mathcal{C}^v$ is used. Each arc of a cycle is visited once per duration of the cycle. Let parameter λ_e^c be the dispatch frequency per hour provided by cycle c on each of its arcs (all arcs of a cycle receive the same frequency of visit from that cycle). Finally, let c_c be the cost (fix + variable) of cycle c. The problem to select the cycles to ensure minimum required dispatch frequency of the arcs of the network takes the following form.

$$
\begin{aligned}
\min \quad & \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{C}^v} c_c y_c && && (2.9)\\
\text{s.t} \quad & \sum_{c \in \mathcal{C}^v} \lambda_e^c y_c \geq z_e^v && \forall e \in \hat{\mathcal{E}}, \forall v \in \mathcal{V} && (2.10)\\
& \sum_{c \in \mathcal{C}^v} y_c \leq N^v && \forall v \in \mathcal{V} && (2.11)\\
& y_c \in \mathcal{Z}^+ && \forall v \in \mathcal{V}, c \in \mathcal{C}^v && (2.12)
\end{aligned}
$$

The objective function (2.9) minimizes the total cost of cycles used to cover the paths. Constraints (2.10) guarantee that the required dispatch frequencies on arcs of each vehicle type are met by the selected cycles. Constraints (2.11) state that the total number of cycles of each vehicle type is restricted by the available fleet size.

2.3.3 Phase 3: Commodity Re-Pathing

This phase aims at revising commodity paths and potentially finding the shortest path for each commodity given the cycle network built in **Phase 2**. The network obtained from the set of selected cycles in **Phase 2** provides a limited capacity on a subset of arcs in $\mathcal{G}^S$. Given the available capacity, one can solve a variant of the IP solved in **Phase 1**, to potentially improve the path selection of the commodities. A path on a cycle network for a specific regime, is considered time feasible if the total length of the path, considering deterministic travel times, transferring times and waiting times, is not greater than the available time to arrive at its destination (service class). To this end, we design the following optimization problem that is solved for each regime.

We propose the following IP-based model in order to choose the unique path per commodity such that the set dispatch frequencies in **Phase 2** are not violated while minimizing the weighted travel time paths of all commodities. Let $\mathcal{E}$ be the set of arcs on the cycle network of the regime considered. Let $\mathcal{P}_k$ denote the set of all time-feasible paths on the cycle network for commodity k. We define t^p as the total length of path p, including transferring and waiting times. Let f^e be the capacity in terms of dispatch frequency on arc e. Let parameter w_k represent the portion of the overall flow (demand rate) that corresponds to commodity k (measured as the percentage of the arrival rate of commodity k over the arrival rate of all commodities). Finally, we define the variable x_k^p, which equals 1 if path $p \in \mathcal{P}_k$ is selected for commodity k, 0 otherwise.

$$\min \quad \sum_{k\in\mathcal{K}} \sum_{p\in P_k} w_k t^p x_k^p \tag{2.13}$$

$$\text{s.t} \quad \sum_{k\in\mathcal{K}} \sum_{p\in P_k} \gamma_e^p q_k x_k^p \leq f^e \quad \forall e \in \mathcal{E} \tag{2.14}$$

$$\sum_{p\in P_k} x_k^p = 1 \qquad \forall k \in \mathcal{K} \tag{2.15}$$

$$x_k^p \in \{0,1\} \qquad \forall p \in P_k, \forall k \in \mathcal{K} \tag{2.16}$$

The objective function (2.13) minimizes the total weighted travel time. Constraints (2.14) guarantee that the dispatch frequency on each arc e is not violated by the flow of all commodity paths using arc e. Constraints (2.15) state that exactly one path per commodity is selected, and restrictions (2.16) indicate the type and domain of variables.

2.4 Post-Processing Phase of Shuttle Scheduling

The commodity paths and shuttle cycles, obtained through the 3-phase approach described in Section 2.3 (referred to as the base plan - BP), are designed based on a flat network. More specifically, constraints (2.6) assume an *ideal* distribution of dispatches along each arc e, i.e., inter-dispatch times are all equal. Given the fact the required dispatch frequency on a given arc is potentially achieved through the coverage of cycles of different lengths, the inter-dispatch times on an arc could vary over time. As a result, an arc covered by multiple cycles may be overly congested during certain periods while not having enough number of dispatches during certain other periods. To remedy this issue, we propose a post-processing phase. The goal of this phase is twofold: (1) to add cycles to the BP to improve on-time performance, and (2) to remove cycles that are not beneficial while ensuring a minimum performance in terms of service level.

2.4.1 Cycle-Adding Phase (CAP)

Based on the BP, we construct a time-space network in order to compute commodities' on-time arrival intervals, which are used to compute an on-time metric for each commodity. We also compute an overall (average) on-time metric, from all commodity on-time metrics, which is weighted by the commodity demand rate q. In Figure 2.2, we show an example of commodity paths from LH1 to LH3, via LH2. The required dispatch frequency over the first arc of the path is provided by Cycle 1, while the required frequency on the second arc of the path is provided by two cycles, named Cycles 2, and 3. The length of these cycles are 0.47, 0.68, and 1.97 hours, respectively. Since the two cycles (Cycle 2 and Cycle 3) covering arc (LH2, LH3) have different lengths, dispatches along this arc are not evenly distributed over time. The construction of the time-space network allows us to identify time intervals during which packages having arrived at the origin would not make it to the destination on time. An on-time execution of a path is depicted in dashed green lines, while an execution of the same path that becomes late is shown in dashed red lines.

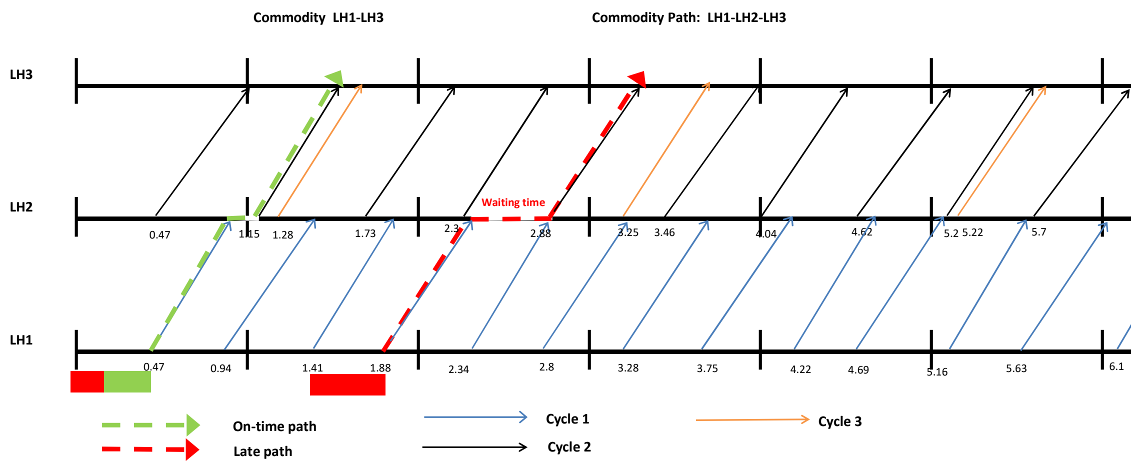

Figure 2.2: Time-Space Network

For a given arrival interval (time between two consecutive dispatches from the origin of the commodity), we first compute the path's travel time starting from the end of the interval to determine whether the commodity arrives on-time to its destination. The service level of the commodity less the length of the path gives the slack or buffer of that specific dispatch. This buffer indicates the maximum waiting time allowed before this specific dispatch from the origin of the commodity while ensuring on-time delivery to the destination, as depicted in the green boxes in Figure 2.3. An on-time metric for each commodity is computed throughout all arrival intervals of the commodity.

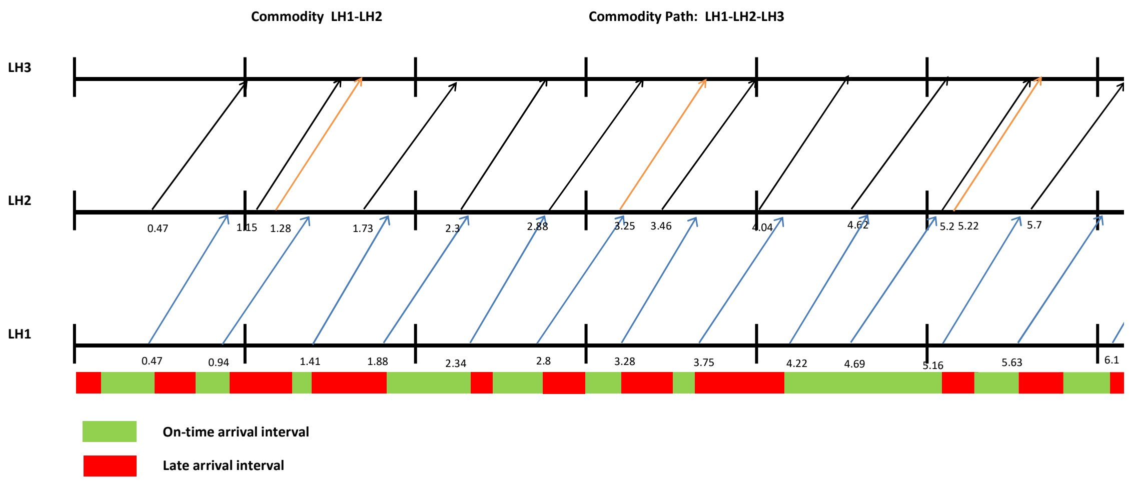

Figure 2.3: On-time and Late Arrival Intervals

We propose a greedy heuristic solution for adding cycles in order to improve the on-time arrival interval metric. For each arc, we calculate a "lateness" score, computed as the sum of all waiting times exceeding the maximum allowed waiting time on each arc of a commodity path. Next, the heuristic selects the edge with the maximum lateness score (in both directions). We design simple two-leg cycles covering the chosen edge. To identify the best start time of the cycle, we test dispatches at discrete start times every t minutes, and the start time improving the on-time metric the most is selected. We add the cycle designed and repeat the procedure until a minimum desired on-time metric is achieved for all commodities and a target overall on-time metric is reached. Detailed description of this approach is provided in Online Supplement A.1.

2.4.2 Cycle-Removal Phase (CRP)

We propose a heuristic approach for the cycle-removal phase. Recall that cycle operation hours of the two regimes overlap for several hours. As a result, removing a cycle from one regime not only affects some commodities in that regime but it may also affect the on-time performance of certain commodities in the other regime. Therefore, we use the set of cycles and commodities from both regimes altogether in this procedure.

The removal of a cycle may have an impact on one or more commodities by either disabling their commodity paths or by deteriorating their on-time performance metric. To quantify this impact, we introduce a metric that represents the loss associated with a disabled commodity based on a monetary measure. First, to estimate this *revenue-loss* measure, we make the assumption that the deterioration of the on-time metric following a cycle removal is a proxy for the number of packages that either arrive later than their due times to their destination or cannot reach their destination anymore. Let us suppose such packages will not generate any revenue (even though they are delivered, but arrived late), and refer to them as zero-revenue packages.

We initially compute the on-time metric for each commodity k when all cycles are considered, denoted by o^k. We also compute the on-time metric for each commodity k when cycle c is tentatively removed, denoted by $o^{k,c}$. The impact of removing cycle c from the cycle schedules on commodity k can be quantified based on $o^k - o^{k,c}$, reflecting the portion of packages of commodity k turning into zero-revenue packages as a result of cycle c being removed. Given the average arrival rate per hour, q^k, for commodity k and the length of the time period where commodity arrivals are considered (i.e, number of hours), denoted by T, one can calculate the number of packages that become zero-revenue packages as a result of cycle c being removed using $T \cdot q^k \cdot (o^{k,c} - o^k)$.

In addition to the revenue associated with each package, one can also calculate the estimated cost of serving each package of a given commodity using a unit cost called the *cost per package per mile*, denoted by ϕ. Unit cost ϕ is calculated as the sum of all cycle costs divided by the weighted sum of commodity arrival rate, q^k and commodity path length, l^k, over all commodities. The cost of a cycle c, denoted by β_c, is defined as the cycle operation cost throughout a day, i.e, the multiplication of the cycle operating hours in a day, denoted by h_c by the variable cost per hour of operating a vehicle of type v, denoted by C_c^v, (depending on the vehicle type of the cycle) plus the fixed cost of operating one of such vehicle, denoted by C_f^v, i.e., $\beta_c = C_f^v + C_c^v h_c$.

Suppose the pricing policy is such that the revenue of serving a package is set to be $\mu\%$ above the cost of serving it. As a result, the loss of revenue from commodity k following the removal of cycle c is obtained as $(1+\mu)\cdot\phi\cdot l^k\cdot T\cdot q^k\cdot(o^{k,c}-o^k)$. A cycle is considered for removal only if the cost of operating the cycle dominates the revenue generated through its operation. Let r_c be the expected return of cycle c, defined as follows:

$$r_c = (1+\mu)\cdot\sum_{k\in\mathcal{K}}\phi\cdot l^k\cdot T\cdot q^k\cdot(o^{k,c}-o^k)-\beta_c \tag{2.17}$$

On the other hand, we impose a minimum standard service level for all commodities, i.e., ensure a minimum on-time metric for all commodities and an overall on-time metric among all commodities (which is equivalent to impose a maximum percentage of commodities that may be late). We also make sure that enough capacity remains available to accommodate commodities when a cycle is removed. Given these financial and operational considerations, a cycle is said admissible for removal if the minimum (worst) on-time metric, minimum overall on-time metric and maximum utilization requirements are not violated when the cycle is removed. Therefore, a cycle c is finally considered for removal if r_c is negative and it is admissible for removal.

The CRP heuristic works as follows. For each cycle, we first verify whether it is admissible for removal. Next, among the cycles that are admissible for removal, we identify the one with the most negative return r_c, denoted by c^*, according to the return function 2.17. Finally, we remove cycle c^* and we keep iterating in a similar fashion until there is no cycle admissible for removal or there is no more cycles with a negative return. The interested reader is referred to Section A.2 for a detailed description of the cycle removal procedure.

2.5 Computational Study

We evaluate the performance of our proposed approach through an extensive computational study based on real-world data provided by our industry partner, one of the leading Chinese courier service providers. The provided data corresponds to historical waybill data for inter-city and intra-city packages over July 2017. For each package, the origin, destination, pickup request time and service class are known. From this information, we compute the average arrival rate per hour of all commodities per regime for the month of July.

The shuttle network is composed of 52 hubs, among which 3 are GHs and 49 are LHs. Considering all possible origin-destination pairs, we have up to 2652 commodities without considering service classes. To reduce the problem size to a more manageable scale, only commodities with an hourly arrival rate q greater than a fixed threshold α are considered for the design of the paths and cycles. That is, commodities with low hourly demand rates ($q < \alpha$) are not directly considered in the process of choosing paths and cycles. Once the set of paths and cycles are identified, commodities with $q < \alpha$ are sent over the available shortest paths.

We allow at most two intermediate stops for the design of commodity paths. Given that most commodities have to arrive at their final destinations or to a GH during the day and in order to smooth out the arrival of packages at GHs, we work with one main service class, 4 hours, for all commodities in the shuttle network. For the creation of cycles, we allow at most four legs and the maximum duration of a cycle is set to 5 hours. Based on the existing fleet of the company, we consider five different vehicle types: 1T, 1.5T, 3.5T, 7T and 14T of capacity.

We experiment with two main settings. First, we design the network considering two regimes, one operating 8am-4pm and the other operating 12pm-8pm, that move both intra-city and inter-city packages. We assume packages arrive in the period 8am-12pm and 12pm-4pm for Regimes 1 and 2, respectively. In this case, the threshold is set to five packages per hour ($\alpha = 5$). Accordingly, commodities with an arrival rate $q \geq \alpha$ represent approximately 90% of the total existing demand. In the second setting, we only consider one regime, operating 8am-8pm focusing on only same-day intra-city packages arriving from 8am-4pm. Here, the threshold is set to two packages per hour ($\alpha = 2$), as the arrival rate volume of same-day delivery packages is significantly lower than intra-city ND plus inter-city packages. We set the target overall on-time performance to 95% and the minimum on-time performance to 85% for both scenarios. For both settings, we perform comprehensive sensitivity analyses.

In what follows, we introduce a series of performance metrics that are used in the subsequent sections to evaluate the obtained results.

2.5.1 Performance Metrics

The main purpose of this section is to define metrics that allow us to measure the quality of the solutions and to compare solutions across different experiments. We define the following main performance metrics making use of the time-space network.

- **On-time:** This metric measures the percentage of on-time arrival intervals of commodities. It captures the percentage of the overall arrival demand in the entire network that is delivered on-time. From the time-space network, we identify all on-time arrival intervals of commodities. Then, we compute a weighted percentage of these on-time arrival intervals over all commodities. The metric is computed as

$$\text{On-time} = \frac{\sum_{k \in \mathcal{K}} o_k q_k}{\sum_{k \in \mathcal{K}} q_k} \tag{2.18}$$

  where o_k is the percentage of on-time arrival intervals of commodity k over the regime horizon.

- **Lateness:** This metric measures how late commodities arrive to their destinations on average. This metric is computed as the weighted average lateness over the late arrival intervals of all commodities. The metric is calculated as follows:

$$\text{Lateness} = \frac{\sum\limits_{k \in \mathcal{K}} l_k q_k}{\sum\limits_{k \in \mathcal{K}} q_k} \tag{2.19}$$

where l_k is the weighted (by the length of late arrival interval) lateness of the late arrival intervals of commodity k over the regime horizon.

- **Earliness:** This metric measures how early commodities arrive to their destinations on average. This is computed as the weighted average earliness over the on-time arrival intervals of the commodities. The metric is computed as folllows:

$$\text{Earliness} = \frac{\sum\limits_{k \in \mathcal{K}} e_k q_k}{\sum\limits_{k \in \mathcal{K}} q_k} \tag{2.20}$$

where e_k is the weighted earliness of the on-time arrival intervals of commodity k over the regime horizon.

We also evaluate the number and type of vehicles designed in each regime. This is an important indicator given the fact that the use of more vehicles is more costly in general. We calculate the utilization of arc measured as the sum of commodity flows passing through the arc divided by the total dispatch frequency on the arc. Finally, we provide statistics on the number of intermediate stops along commodity paths.

2.5.2 Inter- and Intra-city Experiments

In this section, we focus on the two-regime setting moving both intra-city and inter-city packages. For each phase of the approach, we show how the main metrics evolve for BP, after CAP, and after CRP. For the final cycles designed, we present and discuss more detailed results. Additionally, we conduct some sensitivity analysis at the end. Given that the paths and cycles are designed considering only the commodities with $q \geq \alpha = 5$ for this experiment, in Table 2.1, we report statistics separately for such commodities and also for all commodities regardless of their arrival rates.

Table 2.1: Commodity Statistics for Inter-city Experiments

Regime	R1		R2	
Commodities considered	All	$q \geq 5$	All	$q \geq 5$
Number of commodities	2399	723	2350	675
% volume of commodities	100%	98.31%	100%	97.47%
# commodities paths with 0 stops	292	197	301	210
# commodities paths with 1 stop	858	378	938	367
# commodities paths with 2 or more stops	1249	148	1111	98

As we observe from Table 2.1, a significant level of consolidation occurs in both regimes because about 70% of the commodities, equivalent to approximately 50% of the total volume, are taking paths with at least one intermediate stop.

Cycle Adding and Removal Phases

We perform CAP and CRP over the cycles designed for BP. In Table 2.2, we report the on-time metric for each phase. As we see the average on-time performance achieved by BP is already high, close to 100%, even taking into account the commodities with $q <$

α. This is due to the fact that $q \geq \alpha$ represent a high portion of the total commodities; designing paths and cycles based on these commodities guarantees a high average on-time performance. In the subsequent phases, the on-time metric does not change significantly even when removing cycles. Since the on-time metric is already higher than the target on-time metric (95%), the CAP phase mainly aims at bringing up the minimum on-time metric of commodities with $q \geq \alpha$ to at least 85%. On the other hand, CRP identifies and removes cycles that in their absence, neither on-time nor minimum on-time metrics drop below the respective targets and the maximum arc utilization does not exceed 100%.

Table 2.2: On-time Metrics for Inter-city Experiments

Regime	R1		R2	
Commodities considered	All	q ≥ 5	All	q ≥ 5
BP	99.69%	99.96 %	99.49 %	99.93%
CAP	99.78%	99.99%	99.64%	99.99%
CRP	99.66%	99.97%	99.45%	99.96%

In Table 2.3, we show how the minimum on-time metric of commodities with $q \geq \alpha$ improves among the different phases. BP starts with a minimum on-time metric of 57%. CAP improves it until the target of 85% by adding 43 cycles of vehicle type 1T to regimes R1 and R2 (see Table 2.4). Next, CRP removes 72 and 78 cycles of vehicle type 1T from R1 and R2, respectively, without violating the minimum on-time delivery target of commodities with $q \geq \alpha$. Note that since commodities with $q < \alpha$ are not considered in the design, the existence of a path allowing on-time delivery of all such commodities in the network of cycles is not guaranteed. Consequently, the minimum on-time delivery when considering all commodities is zero: there is at least one commodity with $q < \alpha$ and no path guaranteeing its on-time delivery. Due to the low arrival rate of such commodities, policies such as as-needed single dispatches can be added.

Table 2.3: Minimum On-time Metrics for Inter-city Experiments

Regime	R1		R2	
Commodities considered	All	$q \geq 5$	All	$q \geq 5$
BP	0.00%	57.53%	0.00%	57.14%
CAP	0.00%	86.13%	0.00%	85.60%
CRP	0.00%	85.11%	0.00%	85.27%

Table 2.4: Statistics on Number of Cycles at the End of Different Phases

Regime	R1			R2		
Phase	BP	CAP	CRP	BP	CAP	CRP
Number of cycles 1T	130	173	101	148	191	112
Number of cycles 3.5T	177	177	177	170	170	170
Total number of cycles	307	343	278	318	368	282

We now focus on the final schedules at the end of CRP. Table 2.4 indicates that although we allow the model to choose from a wide range of available vehicle capacities, the model tends to only use vehicles of size 1T and 3.5T. Additionally, Table 2.5 shows that the average arc utilization is generally low. This indicates that in most of the cases, the timing requirement constraints on arcs are the main driver for dispatch frequencies, and the capacity requirements are often non-binding constraints. However, the maximum arc utilization is close to 100%, and this mainly happens on arcs destined to or originating from GHs. Since we do not consider commodities with $q < \alpha$ in the design, when we compute the arc utilization metric taking into account all commodities, in some rare cases maximum arc utilization exceeds 100%.

As we observe in Table 2.5, the average arc utilization in R1 is slightly higher than R2.

This can be explained by the fact that the number of commodities as well as commodity volumes in R1 are higher than R2. Additionally, Table 2.4 shows that the proposed approach was able to design a cycle network involving a smaller number of vehicles of each size for R1.

Table 2.5: Arc Utilization in the Final Schedule

Regime	R1		R2	
Commodities considered	All	$q \geq 5$	All	$q \geq 5$
Average	38.9%	34.66%	27.09%	22.02%
Maximum	107.95%	99.99%	104.97%	99.99%

In Figures 2.4 and 2.5, we show the designed networks showing only the arcs with a utilization rate above 90%, for R1 and R2, respectively. For both regimes, we observe that most of the highly utilized arcs are located in the central part of the network. These arcs are often used for the purpose of consolidation and are typically used by several commodities. We also observe a high utilization over arcs originating from or destined to GHs. This can be explained by the fact that the majority of the flow corresponds to inter-city demand which is required to be directed to a GH for sorting activities. Moreover, due to the many-to-few trend of flow of commodities originating from LHs and destined for GHs, we notice that in both regimes, arcs heading to GHs have typically the highest utilization.

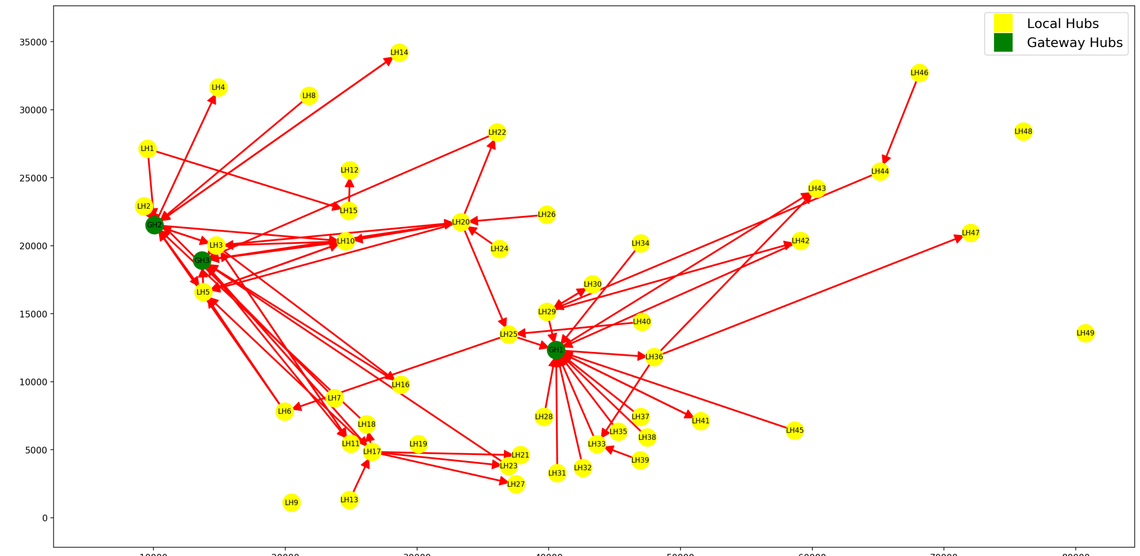

Figure 2.4: Arc utilization over 90% for R1

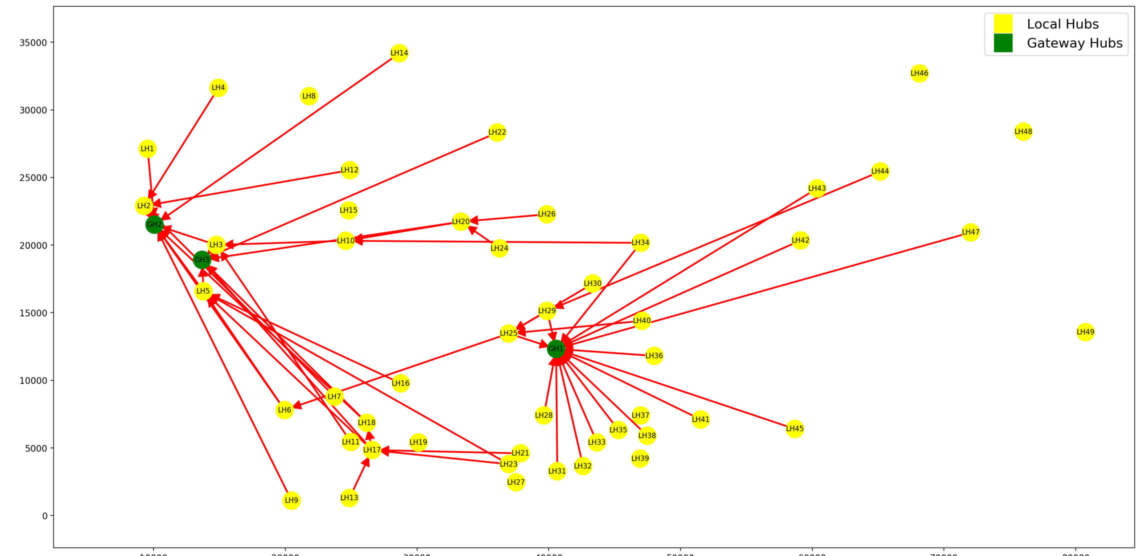

Figure 2.5: Arc utilization over 90% for R2

In Table 2.6, we report results for on-time, earliness and lateness metrics previously defined. We observe that the average on-time metric, which is weighted by commodity volume, is close to 100% for both regimes even when considering all commodities; only less than 1% of the total volume of commodities is late.

On-time commodities with $q \geq \alpha$ arrive to their destination on average 2.4 hours earlier than their due times in both regimes. This create a wide buffer for unforeseen incidents causing delays. On the other hand, late commodities with $q \geq \alpha$ experience a maximum delay of 4 and 7 minutes in R1 and R2, respectively. In many cases, this amount of lateness is not perceived as late from a practical viewpoint.

Table 2.6: Statistics on Time-Space Network for Inter-city Experiments

Regime	R1		R2	
Commodities considered	All	$q \geq 5$	All	$q \geq 5$
On-time	99.66%	99.97%	99.45%	99.96%
Minimum on-time	0%	85.11%	0%	85.27%
Average lateness (hr)	0.15	0.03	0.24	0.06
Worst lateness (hr)	0.30	0.07	0.48	0.12
Average earliness (hr)	2.43	2.45	2.63	2.66
Worst earliness (hr)	2.09	2.11	2.40	2.44

Sensitivity Analysis

In order to measure the robustness of our approach, we study the performance of the designed schedules vis-vis changes in the demand or the available time allocated to shuttle activities.

In the first series of tests, we aim at stress-testing the system by accentuating peaks and valleys in commodity arrival rates. This is done by increasing the demand rate of those commodities with already high demand rates and reducing the demand rates of commodities with low demand rates. These changes are conducted in such a way the overall package volume in the system is kept almost the same. The procedure to create such test sets from the actual data is as follows. We first sort commodities according to their demand rates

q. Next, we select the group of commodities with demand rate between the 40th and 50th percentiles. A second group is selected including commodities with demand between the 50th to 60th percentiles. We then randomly choose a commodity from the first group and a commodity from the second group. We decrease the demand rate of the the first commodity by a given percentage (40% or 80%) of q and add this volume to the second demand rate of the second commodity. This way we ensure total demand is kept unchanged. We do this procedure for all remaining commodities in these groups.

Table 2.7: Modification of Peaks and Valleys for Inter-city Experiments

Regime	R1		R2	
Percentage of change	40%	80%	40%	80%
On-time	99.97%	99.97%	99.96%	99.96%
Minimum on-time	85.11%	85.11%	85.27%	85.27%
Average lateness (hr)	0.06	0.06	0.06	0.06
Average earliness (hr)	2.46	2.46	2.66	2.66
Average utilization of arcs	34.89%	35.12%	21.18%	21.46%
Maximum utilization of arcs	151.94%	151.94%	101.34%	101.34%
Percentage of arcs utilized over 100%	1.79%	1.79%	0.69%	1.04%

Table 2.7 reports main results for percentage changes of valleys and peaks equal to 40% and 80%, respectively. As we observe, the timing metrics including on-time, minimum on-time, average lateness, earliness metrics remain almost unchanged. Moreover, we observe that the average arc utilization does not vary either as the flow is shifted across the network. However, we notice that the maximum arc utilization has increased significantly, and represents the capacity constraint violation along certain arcs. This is more prominent for R1, where at least one arc is utilized more than 50% of its available capacity. We also report the percentage of the arcs with exceeded capacity. We observe that this only occurs in a

small percentage of arcs. In practice this means that some commodities have to wait more for a vehicle with enough capacity on the over-utilized arcs, and therefore they may arrive later than expected to their destinations. On the other hand, for R2, the maximum utilized arc is only 1% over the available capacity which happens in only a small percentage of the arcs as well.

In a second series of tests, we perform a sensitivity analysis to measure the impact of the portion of the available time allocated to shuttle activities vs. those of other activities (e.g., rider activities), in terms of resources needed (number of cycles). In these tests, we increase the time a package can spend in the shuttle system by 30 minutes. To maintain the overall service class unchanged, this can be translated into reducing time available to rider activities at the origin and destination sides by 15 minutes each. The three-phase shuttle scheduling approach (discussed in Section 2.3) is run based on these new parameters, followed by CAP and CRP. Table 2.8 reports the on-time, earliness and lateness metrics on these tests. On-time and lateness metrics are similar to the solutions of the original setting. Not surprisingly, earliness improves approximately 30 minutes for R1 and 26 minutes for R2. The number of cycles needed (see Table 2.9) decrease by 8.6% (from 278 to 254) for R1 and by 2.5% (from 282 to 275) for R2 wherein most of the changes are for cycles of type 1T. An average of 5.5% total decrease in resources is a significant cost reduction.

Table 2.8: Modification of Available Time for Inter-city Experiments

Regime	R1		R2	
Commodities considered	All	$q \geq 5$	All	$q \geq 5$
On-time	99.72%	99.97%	99.57%	99.98%
Average lateness (hr)	0.17	0.04	0.30	0.11
Average earliness (hr)	2.93	2.96	3.08	3.12

Table 2.9: Cycles Designed for Inter-city Experiments

Regime	R1	R2
Number of cycles 1T	78	104
Number of cycles 3.5T	176	171
Total number of cycles	254	275

2.5.3 Intra-city Experiments

In this section, we isolate intra-city same-day activities and solely focus on those. For these experiments, we first present results for commodity path solutions. We then discuss how the on-time metric evolves throughout the different phases. Next, we discuss a more in depth analysis of the results for the final cycles designed. Finally, we conduct a sensitivity analysis. Note that since the overall volume of commodities involving the same-day delivery is significantly lower than the experiments considering all intra- and inter-city commodities, in these tests the design threshold α is set to 2.

Table 2.10 reports statistics on commodity paths. One can observe a high level of consolidation taking place as more than 80% of the commodities, equivalent to approximately 80% of the total volume, are taking paths with at least one intermediate stop. This level of consolidation is higher in this scenario than the intra- and inter-city setting since most of the q's are low (only same-day delivery packages are considered). Therefore, in this setting there is more incentive for consolidation.

Table 2.10: Commodity Statistics for Intra-city Experiments

Commodities considered	All	$q \geq 2$
Number of commodities	2299	1441
% volume of commodities	100%	89.24%
# commodity paths with 0 stops	341	279
# commodity paths with 1 stop	1203	868
# commodity paths with 2 or more stops	755	294

Cycle Adding and Removal Phases

In Table 2.11, we report the on-time and minimum on-time metrics for each phase. We observe that the on-time perfomance is high for both cases with commodities with $q \geq \alpha$ and all commodities.

The obtained on-time metric is already higher than the target (95%), and CAP is mainly to enhance the minimum on-time metric to at least 85%, among the commodities with $q \geq \alpha$. Finally, CRP discards cycles that are not beneficial w.r.t. the target on-time, verifying that maximum utilization of arcs does not violate the available capacity.

Table 2.11: On-time Metrics for Intra-city Experiments

Metric	On-time		Minimum On-time	
Commodities considered	All	$q \geq 2$	All	$q \geq 2$
BP	95.73%	97.15%	0.00%	37.78%
CAP	98.10%	99.24%	0.00%	85.04%
CRP	97.52%	98.89%	0.00%	85.04%

As it can be observed from Table 2.11, the BP involves a minimum on-time metric of

37.78% and 265 cycles type 1T. CAP improves the minimum on-time metric to the target of 85% by adding 56 cycles of vehicle type 1T. Next, CRP removes 54 cycles of vehicle type 1T, resulting in a remaining set of 267 cycles of vehicle type 1T. Similarly to the intra- and inter-city setting, given the fact that commodities with $q < \alpha$ are not considered in the design, the existence of time-feasible paths for such commodities in the network of cycles is not guaranteed. Consequently, the minimum on-time metric of 0% indicates the existence of at least one commodity with $q < \alpha$ and no time-feasible paths.

We now focus on the final schedules designed for the same-day intra-city network. Based on our approach, a total of 267 cycles of vehicle type 1T are required for this setting. The fact that the chosen fleet consists of only 1T vehicles (smallest available trucks) is mainly due to smaller demand rates when considering only same-day intra-city commodities.

Table 2.12: Arc Utilization for Intra-city Experiments

Commodities considered	All	$q \geq 2$
Average	11.64%	9.96%
Maximum	48.91%	48.45%

Figure 2.6 depicts the designed network featuring only the arcs with utilization above 30%. Most of the highest utilized arcs are concentrated around the center of the network, where the most used intermediate stop hubs are located. We do not see a high utilization of arcs in general. The average is around 10% and the maximum approximately 50%. This suggests that smaller vehicle type may be more efficient in terms of making a better utilization of the network, and possibly in the total number of resources needed.

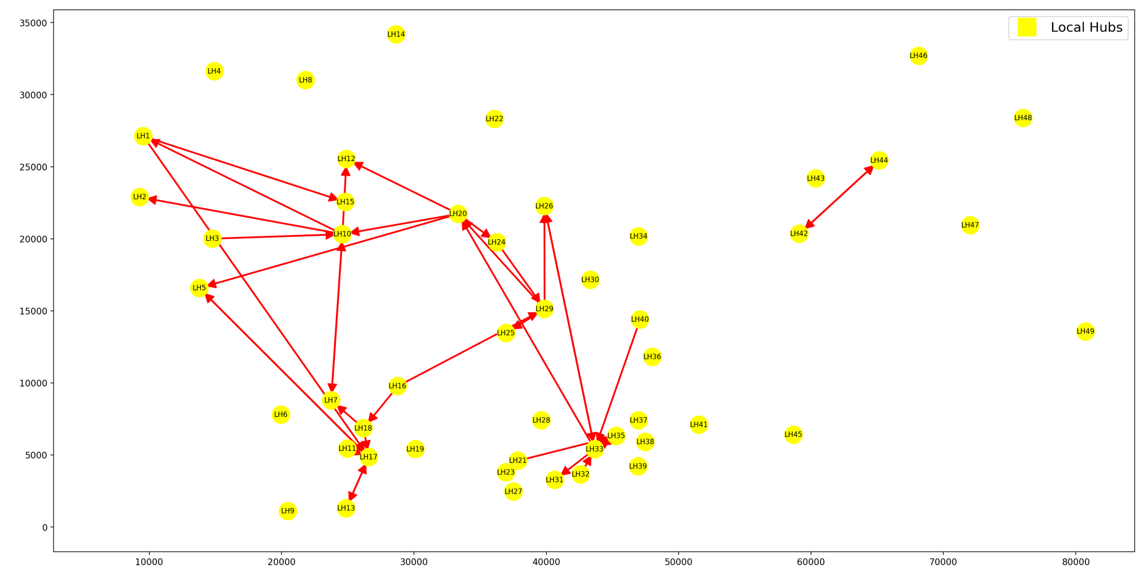

Figure 2.6: Arc Utilization over 30% for Intra-city Experiments

In Table 2.13, we report results for the on-time, earliness and lateness metrics. We see that the on-time metric is close to 100% even when taking into account all commodities. Only around 2% of the total volume of commodities is late.

Commodities with $q \geq \alpha$ arrive on-time to their destination approximately 2 hours earlier on average than the due time, slightly lower when compared to the 2.4 hours in the inter-city setting. Late commodities with $q \geq \alpha$ on the other hand, reach their destination approximately 10 minutes late on average.

Table 2.13: Statistics on Time-Space Network for Intra-city Experiments

Commodities considered	All	$q \geq 2$
On-time	97.53%	98.90%
Minimum on-time	0.00%	85.04%
Average lateness (hr)	0.21	0.16
Worst lateness (hr)	0.42	0.32
Average earliness (hr)	1.84	1.90
Worst earliness (hr)	1.30	1.35

Sensitivity Analysis

We conduct sensitivity analysis, similar to the one performed for the intra- and inter-city scenario. The same procedure for modifying the peaks and valleys in demand rates, and using the same percentage of change is pursued.

Table 2.14 reports the main results for a percentage change of demand rates' valleys and peaks of 40% and 80%, respectively. We see that the on-time metric remains similar and the minimum on-time metric does not change because this indicator does not depend on q. In addition, the average lateness, earliness and arc utilization metrics only vary slightly. Contrary to the case of intra- and inter-city setting, in this scenario, we do not have over utilization of arcs.

Table 2.14: Modification of Peaks and Valleys for Intra-city Experiments

Percentage of change	40%	80%
On-time	98.91%	98.92%
Minimum on-time	85.04%	85.04%
Average lateness (hr)	0.16	0.16
Average earliness (hr)	1.90	1.90
Average utilization of arcs	9.95%	9.78%
Maximum utilization of arcs	49.46%	49.90%

We perform a sensitivity analysis to measure the impact of the available time of commodities on the network. To this end, once again, we suppose that 30 minutes more is allocated to shuttle activities. The three-phase approach as well as CAP and CRP are run. Table 2.15 shows the on-time, earliness, and lateness metrics. On-time and lateness metrics are very similar to the proposed solution. Not surprisingly, earliness improves approximately by 20 minutes given that more time is given to the commodities to reach their destinations. The number of cycles needed decreases from 267 to 216, representing a 19% reduction in required resources. Here, the decrease in the number of cycles is more prominent compared to the inter-city setting. This occurs due to the fact that more available time permits higher consolidation on a network that is under-utilized, and therefore, the impact is more significant in the resources needed when compared to the intra- and inter-city scenario.

Table 2.15: Modification of Available Time for Intra-city Experiments

Commodities considered	All	$q \geq 2$
On-time	98.28%	99.23%
Average lateness (hr)	0.21	0.16
Average earliness (hr)	2.17	2.23

2.6 Conclusion

We propose a multi-phase approach to design static schedules for vehicle movements in a highly dynamic network design system. We propose a heuristic methodology for the design of shuttle schedules that serves intra-city and part of inter-city packages among LHs and GHs. We solve the shuttle schedule in separate subsequent phases: first, we design a single path for each commodity while maximizing consolidation, then we identify the least-cost set of cycles that cover all the paths with required frequency per regime, next we re-path everything over the possible shortest path in the cycle network while respecting the capacity of the network. Finally we add and remove cycles to improve on-time performance and efficiency.

We implemented our methods on a real-world data set provided by our courier service provider partner. We design shuttle schedules that ensure nearly 99% on-time performance and a minimum of 85% on-time performance for the inter-city setting. Most of the commodities arrive earlier to their destinations, around 2 hours in advance, which provides a strong support for unpredictable delays. Among the commodities that are late, most of them are late within 7 minutes on average.

The shuttle schedules proposed are meant to be implementable on a static basis, where same schedules are performed during each day of a week or a fixed horizon time. Different levels of dynamism including settings with static route per day-of-week, re-routing shuttles in response to demand variations at different times of a day, are challenges we aim to address in future work as well as other types of dynamics not considered in this work.

CHAPTER 3
SERVICE NETWORK DESIGN WITH HUB SELECTION

3.1 Introduction

We consider a strategic hub selection problem within the context of service network design for a package courier system operating time-definite services. Time-definite services deliver customer shipments with guaranteed transit times, and faster transit time guarantees may limit consolidation opportunities. Common time-definite package services today include same-day, next-day and two-day services, where packages are collected, routed through a consolidation network of terminals, and then delivered to customers with high velocity; same-day service is typically only provided when both the origin and destination lie within the same urban area. Demand for time-definite package services is growing largely due to growth in e-commerce ([2]).

Package couriers use consolidation terminals to aggregate small shipments into trailerloads or containerloads. They also frequently route packages through intermediate sorting hubs when volumes are smaller for specific origin and destination terminals and when time allows. We will use the following nomenclature to refer to terminals in a package courier system. Consolidation terminals will be labeled *satellites* (or *end-of-line*) facilities and *hub* facilities. Satellites serve only as origin or destination consolidation and sorting points, and they provide the interface between the local pickup-and-delivery (*last-mile*) subsystem and the linehaul (*middle-mile*) subsystem. Hub terminals also provide origin/destination consolidation, but they serve the additional role of intermediate sorting and transfer within the linehaul subsytem.

In this work, we collaborate closely with one of the largest package couriers in China, whose business model includes a plan to grow high-velocity ground transportation (trucking) services between Chinese cities with a new operating model. We construct a strategic optimization approach to select which terminals are the most cost-effective locations to serve as transfer hubs for such systems. The approach is also used to assess the marginal operating cost improvements that result from adding additional hub locations, and thus the approach can be used to determine the right number of transfer hubs for a given system.

Consider a package service network composed in part by a large number of package sorting terminals. Our industry partner denotes these terminals as *gateway hubs*, and there is a typically at least one such terminal serving a region surrounding a city. Larger mega cities may be served by multiple gateway hubs located in different geographic subareas of the city. Gateway hubs receive outbound packages from smaller local pickup-and-delivery hubs within the city, and send inbound packages to these local hubs for final delivery. Gateway hubs enable both small parcel sorting into intermediate containers (like parcel bags) as well as cross-docking of containers and larger parcels. Our industry partner operates terminal buildings with a variety of sizes and capacities. In some cases, not all truck types are permitted at a given gateway hub given parking and dock limitations. In addition, gateway hubs each have a limited number of *loading docks* (doors) that can be used for dispatching outbound trucks to other gateway hubs. The number of outbound loading doors can place limitations on the plans for loading parcels.

Outbound inter-city parcels arriving to a gateway hub are destined for another gateway hub serving the destination region. When cost-effective, such parcels can be loaded into truck trailers traveling *direct* to the destination gateway hub. The alternative to loading direct is to load to an intermediate gateway hub; some parcels may visit multiple intermediate hubs en route. Intermediate sorting, whether parcel sorting or container cross-docking, requires time and depending on the workload of the hub may require minutes or perhaps up to a few hours. Flow and load planning problems are common service network design optimization problems used to make decisions regarding how parcels move through a network of terminals; flow planning problems typically specify a path of intermediate sorting terminals to visit (if any) between origin and destination, and load planning problems additionally determine how to load parcels into different types of trailers (consistent with the flow plan) to be dispatched at specific times.

Some inter-city packages are destined for same-day delivery service in nearby cities, while the remainder of high-velocity packages require next-day delivery or two-day delivery services and may be traveling further. Different types of trucks are used to transport packages. Long-haul inter-city packages are usually transported by medium to large truck types, namely types 14T and 30T, while shorter movements to nearby terminals may be executed with smaller trucks, namely truck types 7T and smaller.

The region we analyze, the southern area of China, is composed of 29 cities with 69 total gateway hubs; see Figure 3.1 for a map. In this region, out industry partner handles 4 million packages on average every day. These packages can be grouped into 120,000 *commodities* identified by origin, destination and service time requirement. Problems of such scale lead to enormously challenging service network design optimization problems. Therefore, the approaches we develop must scale to problems of these sizes. We will see in this chapter that it will be necessary to develop optimization models that can be solved effectively via decomposition approaches to be able to obtain high-quality solutions in a reasonable amount of computational effort.

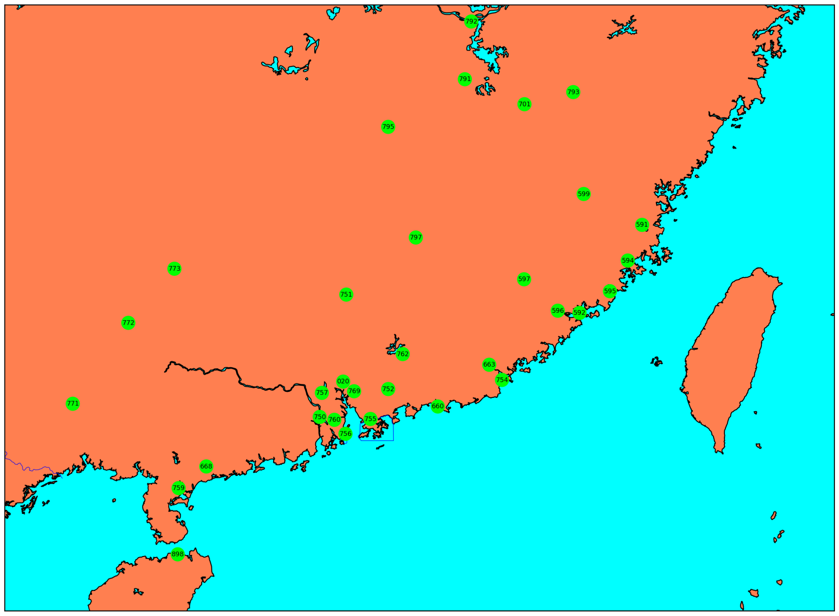

Figure 3.1: Cities in the Southern Region of China

The novelty of this work is that we add an additional layer of complexity on top of the typical flow and load planning problems of service network design: the selection of which terminals to designate as intermediate hubs for intermediate sorting. We call this problem the service network design problem with hub selection (SND-HS). The aim is to select a subset of the terminals to be used as locations for intermediate sorting; the remaining gateway hubs will serve only as origin or destination satellite terminals. Since time-constraints are critical, we explicitly consider the service time requirements of each commodity and seek a selection of intermediate hubs that enable a flow plan that minimizes package movement and sorting costs while meeting all commodity time constraints.

Given that we address a more strategic problem, we model the SND-HS problem on a flat network. Since an integer programming (IP) model with all possible time-feasible paths through all possible intermediate transfer hubs is very large and impossible to solve in a reasonable amount of time, we develop heuristic approaches that greedily add one intermediate hub on each iteration by solving a smaller integer program. We develop three such IP-based greedy heuristic variants: (1) Greedy-Hub, (2) Greedy-Hub-Full which solves larger integer programs each iteration, and (3) GRASP [1]-Hub that randomizes the selection of intermediate hubs.

In each of the IP-based models we develop, we build on a path-based flow planning formulation where a path-based approach is used to easily enforce parcel transfer flow paths to be time-feasible. In the implementation of this approach with our industry partner, we also enforce a practical restriction on the maximum number of intermediate sorts allowed for any particular origin-destination terminal pair. When this maximum is small, a path-based approach can work well since the number of path selection variables does not grow prohibitively large. Computational experiments show that the greedy approach outperforms the IP with all time-feasible one stop paths by a 20% gap difference when selecting and locating 6 or 7 intermediate hubs. The different greedy variants behave relatively similarly in gap performance, showing a maximum gap difference of at most 2% among them, and the Greedy-Hub-Full is the variant that performs the best in most of the cases.

The contributions of our work are summarized as follows:

- We propose a strategic approach to select the most cost-effective hubs for intermediate sorting activities.
- The selection of hubs adds another layer of complexity on top of the traditional flow and load planning service network design problems. We therefore develop decomposition approaches to maintain tractability for these dificult optimization problems.
- We develop sequential approaches that greedily select one intermediate hub per iteration.
- We show that the different greedy variants developed behave comparatively similarly, while they outperform a complete integer programming approach limited to using all time-feasible one-stop paths.

[1] Greedy randomized adaptive search procedure

The remainder of the chapter is structured as follows. In section 3.2, we present a review of relevant prior literature. In section 3.3, we provide a formal description of the problem. In section 3.4, we describe assumptions and restrictions for the problem that arise in practice. In section 3.5, we present the IP-Based hub selection model. In section 3.6, we describe the greedy approach and the variants we develop: Greedy-Hub, Greedy-Hub-Full, GRASP-Hub and the swap procedure we perform at the end of the approach. In section 3.7, we conduct a computational study based on a real world data set. Finally, in section 3.8, we present conclusions and avenues for future research.

3.2 Literature Review

Since we are not aware of research that has extended detailed flow and load planning problems with the selection of intermediate hubs, we briefly review relevant literature on service network design as well as hub location and hub network design problems.

Early research on service network design problems (SNDP) goes back to the work of [34], [9], and [11]. In [11], representative of this early work, the model takes into account many operational constraints, but it does not determine which hubs to employ as intermediate hubs and instead it assumes the role of the hubs and capabilities as fixed. In [35], the authors solve a large scale service network design problem for an express delivery company. The paper focuses on minimizing the movement cost of packages given the hub capabilities.

In [10], the authors address a large-scale air network design problem for an express shipment delivery operation in which they introduced a composite variable formulation, which is an extended formulation that makes use of aggregated variables that provide stronger lower bounds of the linear relaxation compared to conventional approaches.

Excellent paper reviews on service network design can be found in [35] and reviews related to freight transportation in [3] which focused on modeling and mathematical programming approaches. A more recent review, [4], reveals that scant attention has been given to the hub network design problem. The focus has targeted developing efficient techniques to solve large scale problems, focusing on the cost of package movements. We find a survey of network hub location problems in [36], and subsequent works on hub location and the p-hub median problem in [37] and [38]. A review of hub network design [39] considers whether interconnection among hubs should be full or partial, but capacity constraints had not been taken into account at the time of this review. A hub network design work in [40] focuses on finding the optimal set of aircraft routes while minimizing transportation costs but pays little attention to the hub characteristics. In [41], models and algorithms for transportation service network design are presented, but attention is given primarily to the routing of packages and determining which services to include with no focus on where terminals should be located or how many there should be. The work in [42] also reveals that there are numerous model applications that do not consider or integrate logistic hubs.

In a recent paper, [43] focuses on logistic hub location and emphasizes the importance of considering hubs in service network design. Similar works on the hub location problem are found in [44], [45], [46], and there are extensive reviews of these models in [33] and [47]. In a recent work, [20] presents a service network design problem considering hub capabilities as fixed. In a supply chain article, [48] discusses the economic and technological factors, and new delivery systems, like drones and driver-less cars, which will not only shape the design of the hubs but also the whole infrastructure needed for this to happen.

Most service network design literature focuses on flow and load planning, problems of determining how to create effective consolidation plans using existing transfer terminals. For example, [28] develops an integer programming formulation for LTL freight transportation flow and load planning. A similar work for LTL is presented in [29]. In the work presented in [49], a Lagrangian relaxation approach is presented for a stochastic arc based formulation of a restricted SNDP for load planning. Again, however, the transfer terminal locations, capacities, and roles are fixed in advanced.

In a recent article, [31] discusses the challenges and opportunities in city logistics for the future in which it emphasizes the relevance of developing highly dynamic models that could be able to handle real-time information that can be incorporated into models.

3.3 Problem Description

We consider an inter-city service network where packages need to be moved from their origins to their destinations. The network is defined on a graph $\mathcal{G} = (\mathcal{U}, \mathcal{E})$ with node set $\mathcal{U}$ representing terminals and an arc set $\mathcal{E}$ representing (directed) transportation connections between terminals. We assume a distance d_e and a cost c_e on arc e corresponding to a vehicle movement of capacity Q_e. We assume sorting activity can be performed at terminal u if the terminal is selected as an intermediate sorting hub; the time required for sorting at u is t_u^{cross}. We also assume that only a maximum number, s, of terminals may be selected for use as intermediate sorting hubs.

We define the time requirement of a package as the maximum allowable time to reach its destination starting from its origin terminal. We define a commodity k to be the set of shipments sharing the same origin, destination and time requirement. Let $\mathcal{K}$ be the set of all commodities, where a commodity $k \in \mathcal{K}$ is identified by a tuple $(o_k, d_k, \delta_k, q_k)$, in which $o_k \in \mathcal{U}$ is the origin, $d_k \in \mathcal{U}$ is the destination, δ_k is the time requirement and q_k is the commodity volume measured as weight per time to be transferred.

The primary decision problem we consider is to select the terminals to be employed as an intermediate hubs for sorting, such that there exist time-feasible flow paths for all commodities. The optimal set of intermediate hubs results in the minimum cost set of such flow paths, where the total costs include both package movement and sorting costs.

3.4 Restrictions and Practical Assumptions

The problem considered in this chapter is one of strategic design; once selected, intermediate hubs may need to be reconfigured somewhat to handle the package volume and thus should not be changed frequently. Given the strategic nature of the problem, we also approximate actual operating conditions by making a number of assumptions that allow models to be solved more readily. The assumptions we consider are as follows:

- We only allow flow paths with at most T intermediate hub sorts. For computations, T may need to be set quite restrictively when used in hub selection problems. However, in practice, flow paths can be designed with more than T sorts; this will be the subject of the following chapter.
- We furthermore limit the flow paths considered in the hub selection models by not including paths that introduce excessive *circuity*, defined as the ratio between a path's travel time and the direct travel time between the commodity origin and destination. We will use a factor of ρ and not include flow paths with circuity greater than this factor in the models.
- When the origin and destination of a commodity lie within the same city (but at different gateway hubs), we force the commodity to travel direct when the travel time of the direct route is less or equal to a threshold value τ.
- When multiple gateway hub terminals exist in the same city, we only allow one of these locations to be selected as an intermediate hub for sorting.
- Although our industry partner uses multiple truck types to transfer packages between terminals, we determine the cost of a solution by assuming that all packages are transferred with medium-sized trucks (size 7T) on connections with a travel time lower or equal to τ; and we use the largest size truck (size 30T) on connections where the direct travel time is greater than τ.

3.5 IP-Based Hub Selection Model: Path Formulation

We first present an integer programming formulation for the hub selection model. The model builds upon a standard path-based formulation for service network flow planning that uses a flat network representation. The model aims to minimize only the transportation costs needed to transport the packages, such that all commodities arrive on-time to their destinations. Note that while sorting costs are ignored here, it is easy to include them in the objective function if they can be modeled as linear in the path flow variables. The model furthermore uses the assumptions made in Section 3.4.

We define a path p as a sequence of terminals ($u1 = o, u2, ..., uk = d$) that a commodity k follows from its origin to its destination. Intermediate sorting and reloading occurs at each of the terminals u_2 to u_{k-1}. Let $\mathcal{C}$ be the set of all cities where the set of terminals $\mathcal{U}$ are located, and let $\hat{\mathcal{U}}$ be the set of terminals that can be considered as candidates to be intermediate sorting hubs. Let $\mathcal{U}^c$ denote the set of hubs belonging to city c. Let $\mathcal{P}^k$ represent the set of time-feasible paths for commodity k, and define $\mathcal{U}^p$ as the set of intermediate hubs of path p. We also define the following parameters employed in the formulation. Let γ_e^p be 1 if arc e belongs to path p, and 0 otherwise. Let Γ_u^p equal 1 if hub u belongs to path p, and 0 otherwise. Let $\bar{\Gamma}_u^p$ be 1 if hub u is an intermediate stop belonging to path p, and 0 otherwise. Given the set of all time-feasible paths for commodities, we define M_u as the maximum number of commodities that can possibly have paths with an intermediate stop at u and M_e as the maximum number of commodities that can possibly have paths containing arc e. Let f_e be the available remaining truck capacity on arc e; recall that some commodities may have paths fixed in advance, for example the intra-city commodities forced to take the direct path.

We now define the decision variables to be employed in the formulation. Let binary variable x_k^p equal 1 if path p is selected for commodity k, and 0 otherwise. Let integer variable y_e be the number of vehicles used on arc e. Also, let binary variable w_u be 1 if terminal u is selected as an intermediate sorting hub, and 0 otherwise. Finally, let binary variable v_e equal 1 if arc e is used, and 0 otherwise. The model for selecting one time-feasible path per commodity such that package movement costs are minimized is as follows.

$$\min \quad \sum_{e \in \mathcal{E}} y_e c_e \tag{3.1}$$

$$\text{s.t} \quad \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}^k} \gamma_e^p q_k x_k^p \leq f_e + Q_e \cdot y_e \qquad \forall e \in \mathcal{E} \tag{3.2}$$

$$x_k^p \leq w_u \qquad \forall k \in \mathcal{K}, \forall p \in \mathcal{P}^k, \forall u \in \mathcal{U}^p \tag{3.3}$$

$$\sum_{u \in \hat{\mathcal{U}}} w_u \leq s \tag{3.4}$$

$$\sum_{u \in \mathcal{U}^c} w_u \leq 1 \qquad \forall c \in \mathcal{C} \tag{3.5}$$

$$x_k^p \leq v_e \qquad \forall k \in \mathcal{K}, \forall p \in \mathcal{P}^k, \forall e \in p \tag{3.6}$$

$$\sum_{e \in \delta^+(u)} v_e \leq l_u \qquad \forall u \in \mathcal{U} \tag{3.7}$$

$$\sum_{p \in \mathcal{P}^k} x_k^p = 1 \qquad \forall k \in \mathcal{K} \tag{3.8}$$

$$x_k^p \in \{0, 1\} \qquad \forall k \in \mathcal{K}, \forall p \in \mathcal{P}^k \tag{3.9}$$

$$y_e \in \mathcal{Z}^+ \qquad \forall e \in \mathcal{E} \tag{3.10}$$

$$v_e \in [0, 1] \qquad \forall e \in \mathcal{E} \tag{3.11}$$

$$w_u \in [0, 1] \qquad \forall u \in \hat{\mathcal{U}} \tag{3.12}$$

The objective function (3.1) minimizes package movement costs on arcs. In constraints (3.2), commodity path flows force the number of vehicles needed on arc e. Constraints (3.3) verify whether hub u is used as an intermediate stop by any commodity. Constraint (3.4) limits the number of hubs used as an intermediate stop. Constraints (3.5) restrict the selection of at most one intermediate hub per city. Constraints (3.6) verify whether arc e is used by any commodity path. Constraints (3.7) limit the number of different outbound destinations at each hub u. Constraints (3.8) state that exactly one path per commodity is selected. Finally, constraints (3.9), (3.10), (3.11) and (3.12) indicate the type and domain of the variables.

We introduce a second version of the model that employs aggregated versions of the forcing constraints (3.3) which ensure that paths containing intermediate sorts are only used when their sorting hubs are selected and constraints (3.6) related to whether arc e is used in the solution. We use the same sets, parameters and variables defined for the formulation (3.1)-(3.12), and the interpretations of the other constraints remains unchanged. The aggregated formulation is as follows:

$$\min \quad \sum_{e\in\mathcal{E}} y_e c_e \tag{3.13}$$

$$\text{s.t} \quad \sum_{k\in\mathcal{K}}\sum_{p\in\mathcal{P}^k} \gamma_e^p q_k x_k^p \le f_e + Q_e \cdot y_e \qquad \forall e \in \mathcal{E} \tag{3.14}$$

$$\sum_{k\in\mathcal{K}}\sum_{p\in\mathcal{P}^k} \bar{\Gamma}_u^p x_k^p \le M_u \cdot w_u \qquad \forall u \in \hat{\mathcal{U}} \tag{3.15}$$

$$\sum_{u\in\hat{\mathcal{U}}} w_u \le s \tag{3.16}$$

$$\sum_{u\in\mathcal{U}^c} w_u \le 1 \qquad \forall c \in \mathcal{C} \tag{3.17}$$

$$\sum_{k\in\mathcal{K}}\sum_{p\in\mathcal{P}^k} \gamma_e^p x_k^p \le M_e v_e \qquad \forall e \in \mathcal{E} \tag{3.18}$$

$$\sum_{e\in\delta^+(u)} v_e \le l_u \qquad \forall u \in \mathcal{U} \tag{3.19}$$

$$\sum_{p\in\mathcal{P}^k} x_k^p = 1 \qquad \forall k \in \mathcal{K} \tag{3.20}$$

$$x_k^p \in \{0,1\} \qquad \forall k \in \mathcal{K}, \forall p \in \mathcal{P}^k \tag{3.21}$$

$$y_e \in \mathcal{Z}^+ \qquad \forall e \in \mathcal{E} \tag{3.22}$$

$$v_e \in [0,1] \qquad \forall e \in \mathcal{E} \tag{3.23}$$

$$w_u \in \{0,1\} \qquad \forall u \in \hat{\mathcal{U}} \tag{3.24}$$

3.6 Greedy Heuristic Approach

Solving the models of the previous section exactly is likely to be difficult for larger problem instances; we will demonstrate the behavior of this model as problem sizes grow larger in the computational study. Therefore, in this section, we present a greedy methodology that selects terminals to use as intermediate sorting hubs in a sequential fashion. In each iteration of this approach, a single new intermediate sorting hub is added to a set of existing selected intermediate hubs to reduce problem complexity. We propose a number of different variants for how we perform the greedy selection procedure.

The aim here is to reduce the problem of finding a good set of intermediate hubs to one where smaller integer programs can be solved at each iteration in the heuristic search for good-quality solutions in a reasonable amount of time compared to the IP-based hub selection model presented in section 3.5.

In Figure 3.2, we illustrate how the algorithm works at a high level. First, we start with the direct path solution in the first iteration; these paths must be time-feasible, otherwise no solution that adds circuity and intermediate sorting can reduce the path time. We then search for the first intermediate sorting hub to add, among the admissible set of intermediate hubs, that leads to a decrease in total (transportation) costs. One simple way to find an intermediate hub that reduces package movement costs is to solve a flow planning model with fixed hubs; we detail this idea in subsection 3.6.1, and provide a number of solution approach variants. Then, after we select the new intermediate sorting hub, we fix the new subset of intermediate hubs and repeat the process. We propose to keep iterating in the same fashion until we either reach the maximum number of intermediate hubs to select, s, or until we find that solution does not improve by a reasonable amount any more.

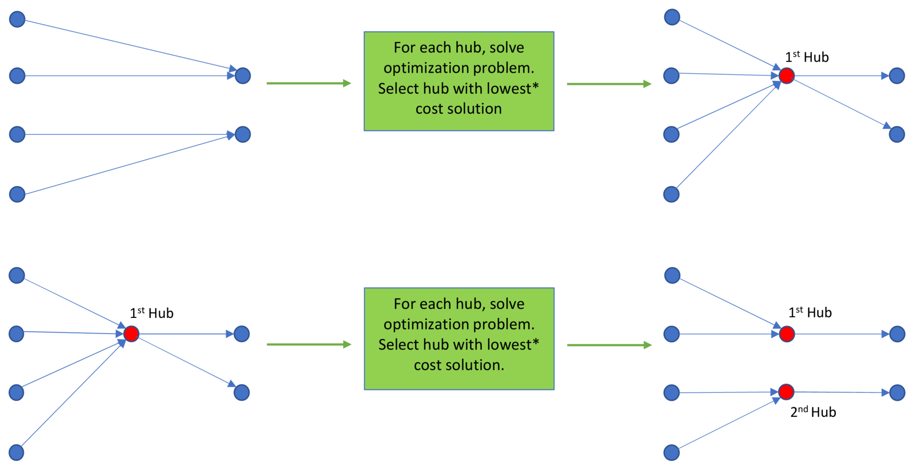
For each hub, solve optimization problem. Select hub with lowest* cost solution
1st Hub
1st Hub
For each hub, solve optimization problem. Select hub with lowest* cost solution.
1st Hub
2nd Hub

Figure 3.2: Illustration of the Greedy Heuristic Approach

Algorithm 1 Greedy Algorithm

```
1: $\mathcal{S} = \emptyset$
2: Set initial path as the direct route for each commodity
3: Set $\pi$ as the cost of the direct solution
4: while $|\mathcal{S}| < s$ do
5:     for $u \in \hat{\mathcal{U}}$ do
6:         Solve flow planning model for $u$ according to the greedy variant employed.
7:         Save solution for each commodity $k$ when evaluating new intermediate hub $u$,
           call cost solution $\pi^u$
8:     end for
9:     Call $u^*, \pi^{u^*}$ the new intermediate hub selected and the updated solution cost re-
       spectively.
10:    $\pi \leftarrow \pi^{u^*}$
11:    Update commodity paths chosen for the corresponding $u^*$
12:    $\mathcal{S} \leftarrow \mathcal{S} \cup u^*$
13:    $\hat{\mathcal{U}} \leftarrow \hat{\mathcal{U}} \setminus u^*$
14:    Update $\hat{\mathcal{U}}$, remove all hubs belonging to the city of the hub $u^*$ incorporated
15:    if $it\%t$ then
16:        Solve flow planning model using the existing set of intermediate hubs $\mathcal{S}$. Set
           time limit $TL$ and gap $GP$
17:    end if
18: end while
19: Solve flow planning model using the final set of intermediate hubs selected, $\mathcal{S}$. Set
    time limit $TL^f$ and gap $0\%$
```

There are a number of ways to implement this basic greedy paradigm. The primary approach that we employ is denoted the Greedy Algorithm and is described in detailed in Algorithm 1. In this approach, we solve a separate flow planning model that adds a single new terminal u to the set of intermediate hubs. After solving this set of optimization problems (and different greedy variants use slightly different assumptions about the available flow paths in this step), a single new intermediate hub u^* is selected to be added to the current solution.

Covering the steps in detail, we first set the direct path solution for all commodities and let π be the solution cost (line 2 and 3). Next, we verify whether we can add more new intermediate hubs (line 4). Then, we iterate over each of the new possible intermediate hubs in set $\hat{\mathcal{U}}$. For each new intermediate hub $u \in \hat{\mathcal{U}}$, we solve a flow planning model (line 7). We save the solution cost π^u and the commodity paths solution. After that, we add the new intermediate hub selected according to the greedy variant employed (line 9). We update the current solution cost and the commodity paths solution (from line 10 to 12). We then remove from set $\hat{\mathcal{U}}$ the new intermediate hub added and all hubs belonging to the city of the new hub. We solve a larger flow planning model every t iterations, with a short time limit, including all time-feasible direct and one-stop paths using all existing intermediate hubs (line 16). Finally, after the maximum number of intermediate hubs is reached, we solve the flow planning model including all paths associated with the final set of intermediate hubs, allowing a longer computation time (line 19).

Since we implement this paradigm in different ways, we now describe each of the variants in more detail:

Greedy-Hub

In this greedy variant, we solve the flow planning model for each new intermediate hub $u \in \hat{\mathcal{U}}$ where the path set includes the paths used in the best solution with the existing intermediate hubs, plus all time-feasible one-stop paths using the new intermediate hub. We select the intermediate hub to add that yields the lowest solution cost.

GRASP-Hub

This variant works similarly to the Greedy-Hub variant, but we randomize the selection of the new intermediate hub among the R best solutions (lowest cost) in each iteration, in which the set $\mathcal{R}$ denotes the set of the R lowest cost new intermediate hubs. This variant selects the new intermediate hub u with probability p_u, which is defined as the inverse solution cost associated with the new intermediate hub u, denoted by c_u, divided by the sum of all inverse solution costs associated with each new intermediate hub $u \in \mathcal{R}$; specifically:

$$p_u = \frac{\frac{1}{c_u}}{\sum\limits_{u' \in \mathcal{R}} \frac{1}{c_{u'}}} \tag{3.25}$$

Greedy-Hub Full

This variant solves the flow planning model for each new intermediate hub $u \in \hat{\mathcal{U}}$, allowing all time-feasible one-stop paths associated with all existing intermediate hubs and the paths with the new intermediate hub u. The variant chooses the new intermediate hub that returns the lowest solution cost.

3.6.1 Flow Planning Model with Fixed Hubs

We introduce the flow planning model with fixed hubs employed in all variants of the greedy heuristic. This formulation aims to decide the unique time-feasible path for each commodity. The model is similar to the hub selection model described in section 3.5, however, it is a more compact formulation since many of the restrictions (such as the at most one intermediate hub per city and the maximum number of intermediate hubs constraints), are already embedded in the greedy algorithm. Thus, we employ the same sets and parameters defined in section 3.5. Likewise, let binary variable x_k^p equal 1 if path p is selected for commodity k, and 0 otherwise, and let integer variable y_e be the number of vehicles used on arc e. The flow planning model used in the greedy variants takes the following form.

$$\min \quad \sum_{e\in\mathcal{E}} y_e c_e \tag{3.26}$$

$$\text{s.t} \quad \sum_{k\in\mathcal{K}} \sum_{p\in\mathcal{P}^k} \gamma_e^p q_k x_k^p \leq f_e + Q_e \cdot y_e \qquad \forall e \in \mathcal{E} \tag{3.27}$$

$$\sum_{p\in\mathcal{P}^k} x_k^p = 1 \qquad \forall k \in \mathcal{K} \tag{3.28}$$

$$x_k^p \in \{0,1\} \qquad \forall k \in \mathcal{K}, \forall p \in \mathcal{P}^k \tag{3.29}$$

$$y_e \in \mathcal{Z}^+ \qquad \forall e \in \mathcal{E} \tag{3.30}$$

The objective function (3.26) minimizes the package movement costs on arcs. Constraints (3.27) force the number of vehicles needed on arc e. Constraints (3.28) state that exactly one path per commodity is selected. Finally, constraints (3.29) and (3.30) indicate the type and domain of the variables.

3.6.2 Swap Procedure

At the end of the greedy procedure, we perform a simple swap heuristic that aims to improve the final solution of the greedy heuristic. To do so, we define a group of cities where swaps are permitted to take place. In this procedure, an existing intermediate hub is only allowed to be swapped for another new hub if both hubs belong to the same cluster. For our primary computational study, we define four clusters that are essentially grouped according to geographical closeness. We illustrate these example city clusters in Figure 3.3 distinguished by four different colors. The swap procedure is described in algorithm 2.

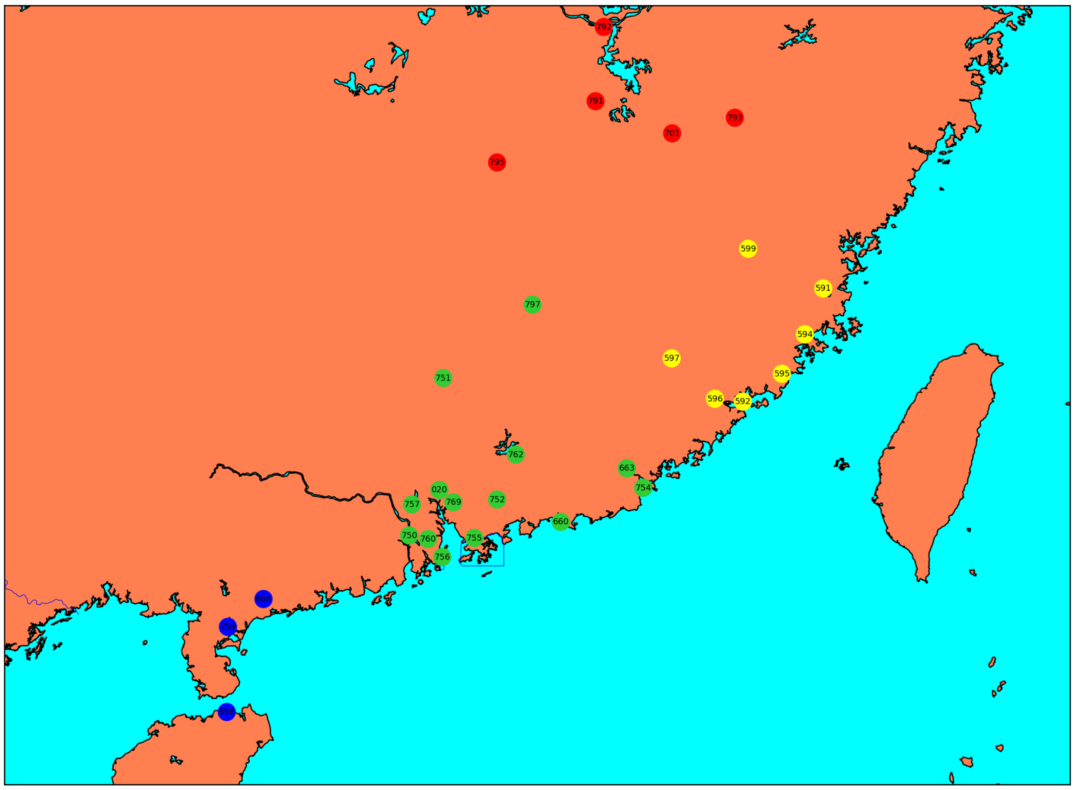

Figure 3.3: Cluster of Cities Differentiated by Closeness

Algorithm 2 Swap Procedure

```
1: Let $\mathcal{U}^{in}$ be the final set of intermediate hubs selected in the greedy procedure
2: $\mathcal{S} \leftarrow \mathcal{U} \setminus \mathcal{U}^{in}$
3: Set initial commodity paths solution and solution cost, $\pi^*$, as the the final solution from the greedy procedure
4: while $n < n^{max}$ do
5:     for $h \in \mathcal{S}$ do
6:         for $u \in \mathcal{U}^{in}$ do
7:             Solve a flow planning model dropping the paths associated with intermediate hub $u$ and adding the paths for intermediate hub $h$; call this solution cost $\pi^u$
8:             if $\pi^u < \pi^*$ then
9:                 Save solution for each commodity $k$ when evaluating swapping hub $u$ for hub $h$
10:                $\pi^* \leftarrow \pi^u$
11:            end if
12:        end for
13:        Select $u^* = \underset{u \in \mathcal{U}}{\operatorname{argmin}} \pi^u$
14:        Swap new intermediate hub $h$ with existing intermediate 'hub $u^*$ (if exists) and update the new solution
15:        $\mathcal{U}^{in} \leftarrow \mathcal{U} \cup h$
16:        $\mathcal{U}^{in} \leftarrow \mathcal{U} \setminus u^*$
17:    end for
18:    $\mathcal{S} \leftarrow \mathcal{U} \setminus \mathcal{U}^{in}$
19:    $n \leftarrow n + 1$
20: end while
```

The core of the swap algorithm 2 works as follows. For each new hub $u \in \mathcal{S}$, we evaluate for each existing intermediate hub $u \in \mathcal{U}^{in}$ whether it is an improved solution cost to swap new hub h for the existing intermediate hub u. If a swap occurs, we save the solution as the new incumbent, and we update the sets $\mathcal{S}$ and $\mathcal{U}^{in}$. We iterate until a maximum number of iterations is reached.

3.7 Computational Study

We test our approaches presented in the preceding sections by conducting a computational study based on real-world data provided by our Chinese courier company partner. For the experiments, we work with the way-bill of inter-city and intra-city packages for April 2019. From this data, we compute an average weight per day and an average number of parcels per day for each commodity.

The network is composed of 69 hubs. Taking into account all possible origin-destination pairs, we may have up to 4692 commodities without considering time requirements. Since we address a strategic design problem, we further simplify the problem by grouping commodities according to different groups of time requirements which are described in subsection 3.7.1. We also reduce the number of commodities by only including commodities that contribute to 99% of the total package weight; since we account for most of the volume in this approach, we believe that the most cost-effective intermediate hubs for sorting will be selected. We use two different vehicle types: 7T and 30T, we set the τ parameter as 1.5 hours and we set the use of paths with at most one stop, i.e. $T = 1$.

In initial experiments, we test many different values of s ranging from 5 to 60. These experiments suggest that for values of s greater than 10, the solution cost does not improve significantly (see B.1). Therefore, we focus on presenting experiments for values of s ranging from 1 to 8. Furthermore, in initial experiments, we considered including a restriction of a maximum number of outbound destinations at hubs; however, the values used in the experiments made the problem infeasible and therefore we had to increase the actual value at least a few times (See B.1). Thus, we do not include this restriction in the experiments we present, and instead we report the violation of this constraint. We test the different greedy variants described in the previous sections, and we run the hub selection model in order to be able to compare gap performance across the different greedy variants. We employ the aggregated formulation of the IP-based hub selection model because it has drastically fewer constraints, given the size of the problem, and also yields significantly lower solution costs compared to the unaggregated version of the model.

3.7.1 Consolidation of Commodities

In this section, we describe how commodities are grouped according to similar time requirements.

Most of the package volume corresponds to intra-city packages (of the southern area of China); in Figure 3.4, we depict the volume percentage (in terms of weight) for both intra-city and inter-city demand.

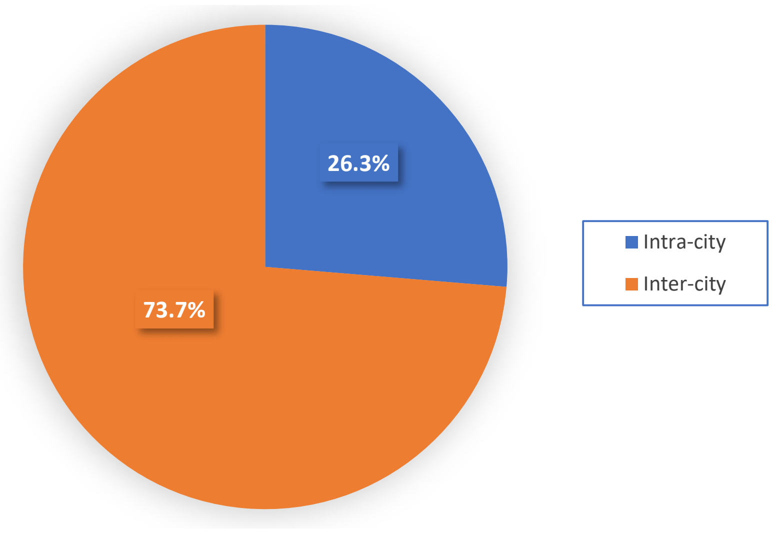

Figure 3.4: Package Volume by Type of Demand

In Figure 3.5, we illustrate the volume of shipments by different time requirements (in hours).

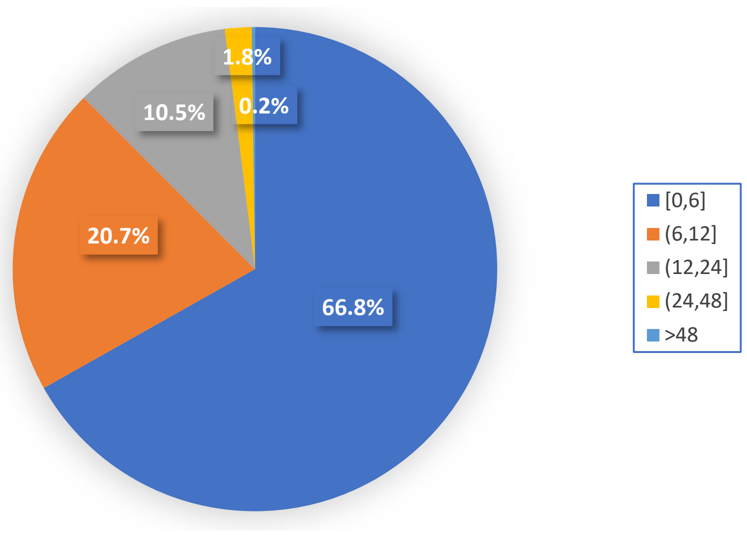

Figure 3.5: Volume of Packages by Time Requirement

We group commodities by employing aggregated time requirement groups, namely intervals, between the bounds of 6, 12, 18, 24, 48 and 72 hours for inter-city packages. Due to this grouping, we are able to decrease the total number of commodities from roughly 120,000 to 5,297. In Table 3.1, we report statistics for this grouped set of commodities.

Table 3.1: Aggregated Commodities by Time Requirement

# of commodities (O-D-δ)	5297
# of O-D pairs	1864
Average # of services per O-D	2.84
Minimum # of services per O-D	1
Maximum # of services per O-D	42

3.7.2 Experiments for Different Values of s

Table 3.2: Gap Performance of the Greedy Variants

s	0	1	2	3	4	5	6	7	8
Hub selection model	NA	NA	30.38%	22.48%	29.02%	32.51%	35.09%	36.10%	32.70%
Greedy-Hub	0.00%	0.00%	24.01%	19.68%	18.02%	17.07%	16.57%	15.76%	15.90%
Greedy-Hub-Full	NA	NA	23.58%	19.30%	17.52%	16.33%	15.72%	15.49%	14.92%
GRASP-Hub	NA	NA	23.46%	18.75%	17.55%	16.99%	16.09%	16.13%	15.39%

Table 3.2 shows the gap performance for the different greedy variants and the hub selection model. The cost lower bound used to compute all gaps is determined by running the full hub selection model. As we observe, the hub selection model does not yield good gap performance after 16 hours of running time. This result is explained by the fact that this model allows all time-feasible one-stop paths associated with the admissible set of all intermediate hubs, resulting in more than 100,000 commodity paths. For the case when $s = 0, 1$, we are able to solve the instances optimally because for $s = 0$, it is trivial to evaluate the direct route for all commodities and for the case $s = 1$, we employ the Greedy-Hub variant which returns the best solution when we evaluate the incorporation of each new intermediate hub, and we then select the hub with the maximum savings. Since these models are small in size (we only have the direct path and the path passing through the new intermediate hub), we are able to solve each flow planning model with fixed hubs exactly. For the cases when $s \geq 2$, the three variants behave comparatively similar in terms of gap performance. For most of the values of s, except for $s = 2, 3$, the Greedy-Hub-Full is the variant that performs the best. This result is not surprising because this variant uses more path alternatives in each iteration, considering all paths associated with all the existing intermediate hubs when solving the flow planning model.

Table 3.3: Number of Hubs Violating the Maximum Number of Loading Docks

s	0	1	2	3	4	5	6	7	8
Hub selection model	NA	NA	4	3	3	2	3	2	3
Greedy-Hub	15	4	2	2	3	3	4	2	3
Greedy-Hub-Full	NA	NA	2	2	2	2	2	2	2
GRASP-Hub	NA	NA	3	2	2	3	2	3	2

Table 3.3 shows the number of hubs that violate the maximum number of loading docks. As we observe, for the most interesting cases, $s \geq 1$, this constraint is violated in no more than four hubs. Moreover, most of these hubs have less than ten loading docks, some having even between 3 and 4. This suggests that it may be beneficial to incorporate more loading docks in these hubs given that they are cost-effective for intermediate sorting.

Table 3.4: Commodity Volume Taking Non-direct Paths

s	0	1	2	3	4	5	6	7	8
Hub selection model	NA	NA	29.1%	30.8%	29.7%	28.8%	27.2%	26.5%	27.8%
Greedy-Hub	0.0%	24.6%	27.0%	27.8%	30.2%	39.0%	38.6%	38.3%	38.0%
Greedy-Hub-Full	NA	NA	25.0%	25.9%	25.5%	34.1%	35.7%	35.7%	35.1%
GRASP-Hub	NA	NA	28.9%	29.1%	26.0%	34.3%	35.8%	39.2%	35.8%

In Table 3.4, we report the degree of consolidation, commodity volume taking non-direct paths, achieved by the different greedy solutions. We observe that the highest percentage of consolidation is close to 40% for the GRASP-Hub and Greedy-Hub variants. This level of consolidation may not be perceived as high, but the vast majority of the intra-city volume, roughly 26% of total volume, is forced to be sent on the direct path and thus consolidation is only possible for the other three-quarters of the total flow volume.

Table 3.5: Inter-city Commodity Volume Taking Non-direct Paths

s	0	1	2	3	4	5	6	7	8
Hub selection model	NA	NA	39.64%	41.94%	40.43%	39.19%	37.04%	36.03%	37.80%
Greedy-Hub	0	33.50%	36.72%	37.77%	41.05%	53.04%	52.51%	52.14%	51.74%
Greedy-Hub-Full	NA	NA	34.00%	35.27%	34.69%	46.45%	48.53%	48.54%	47.79%
GRASP-Hub	NA	NA	39.32%	39.63%	35.38%	46.69%	48.66%	53.33%	48.69%

In Table 3.5, we report the level of consolidation when only considering inter-city demand. As we notice, the degree of consolidation achieved by the different variants is close to 50%, and greater in some cases. As we increase s, we generally observe a higher level of consolidation, which is not surprising given that more paths become available as more new intermediate hubs are incorporated.

In Figure 3.6, we illustrate the geographical location of the cities associated with the intermediate hubs selected by the Greedy-Hub-Full variant, which is the variant that performs the best in most cases. At the top of Figure 3.6, we show the solution for $s = 1, 2$, then going down for $s = 3, 4$, and so on.

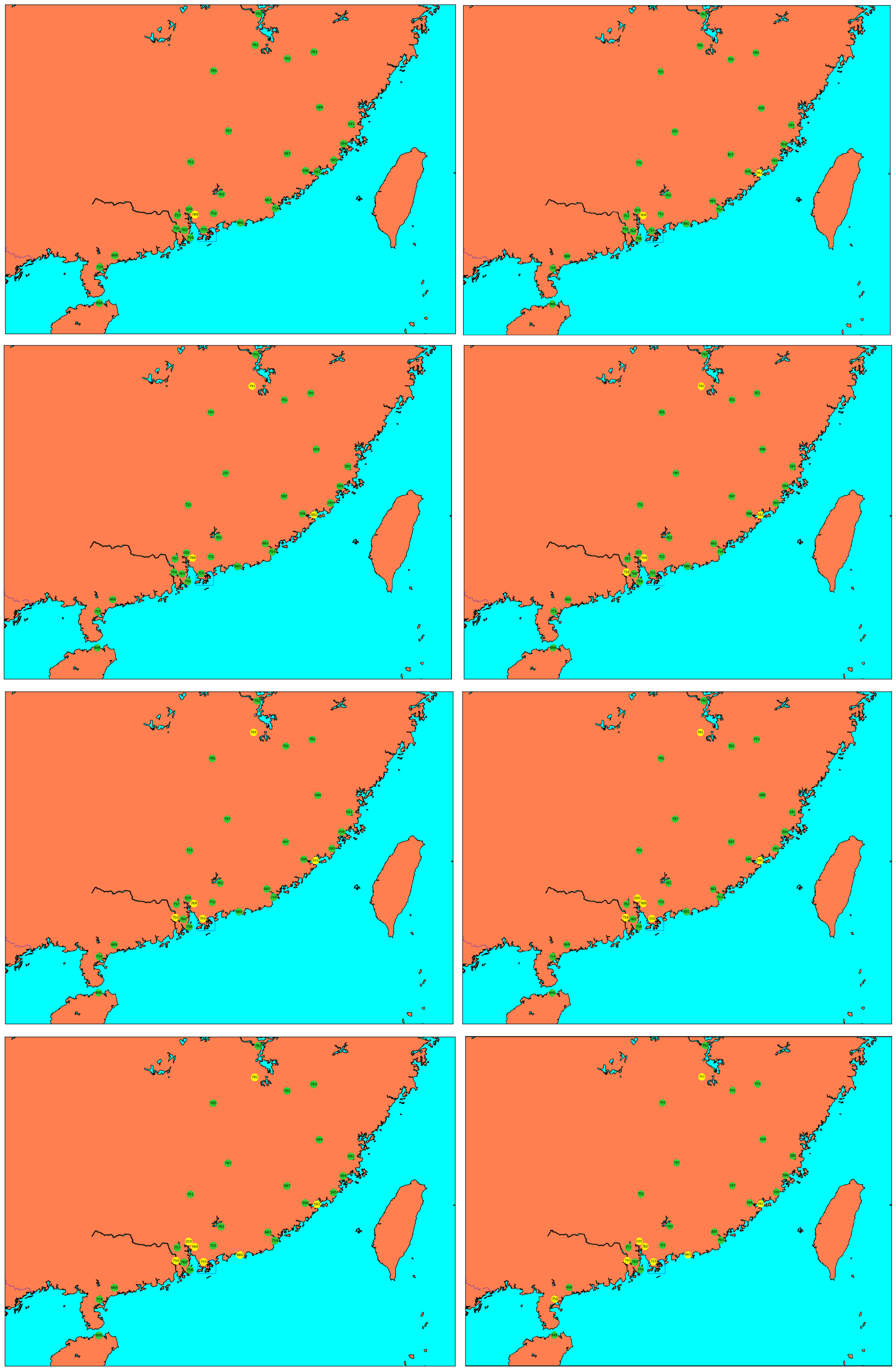

Figure 3.6: Cities of the Intermediate Hubs Selected in the Greedy-Hub-Full Variant

In Table 3.6, we report the percentage of commodity volume sorted at the intermediate hubs selected for $s = 8$. Given that most of the inter-city volume in this data set comes from and heads to hubs located in the center, we notice that the hubs that are more frequently used for sorting are situated around the center. In Figure 3.6, we markedly observe this behaviour from $s = 4$ to $s = 7$. At least 2% of the commodity volume is sorted at each of the intermediate hubs selected. Note that 2% may seem negligible, but this company transfers millions of packages daily. In Figure 3.7, we depict the commodity volume sorted at intermediate hubs; the larger the circle, the greater the commodity volume sorted at the hub. We observe that three hubs concentrate more than 70% of the sorted volume. These hubs are located at the center of the geographical area, which is expected, because most of the package volume is coming from and going to cities situated around the center.

Table 3.6: Commodity Volume Cross-docked at Intermediate Hubs

Hub	% Sorted Volume
755WE	37%
769WB	26%
592W	9%
750W	9%
020R	8%
660VW	6%
791WH	4%
759VA	2%

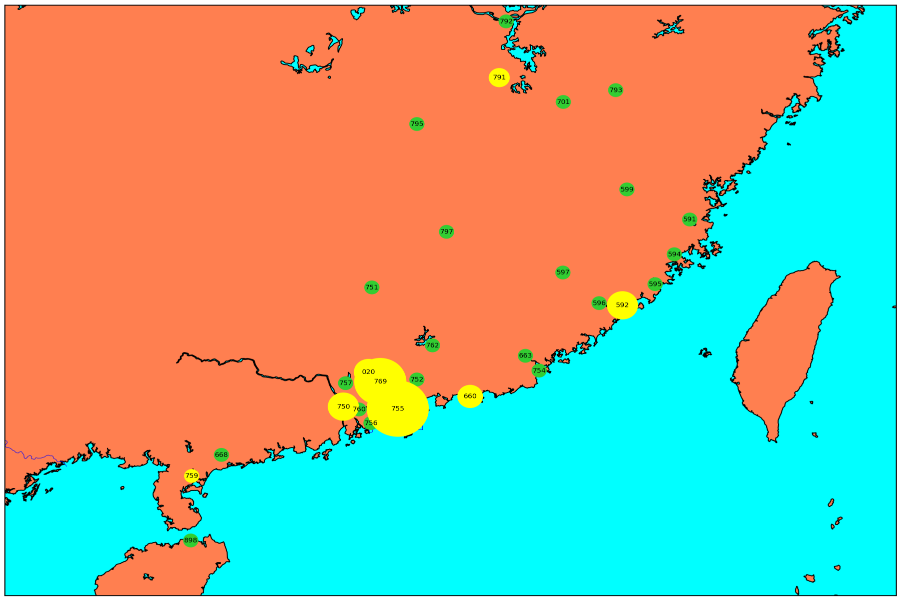

Figure 3.7: Cross-docked Volume at Intermediate Hubs

3.7.3 Experiments Performing Swap Procedure

We perform a swap procedure at the end of the Greedy-Hub-Full variant. Swaps are only allowed among hubs belonging to the same city cluster defined in the subsection 3.6.2.

In Table 3.7, we report the number of swaps that the method performs after there is no change in solution cost in swapping existing intermediate hubs with new intermediate hubs. We observe that only for $s = 2, 3$, a swap occurs, and the swaps take place in the same cities. Moreover, the gap improvement of the swap scheme upon the greedy variant is not greater than 1%. This suggests that an enhanced swap procedure needs to be developed to improve current greedy solutions.

Table 3.7: Statistics when Performing Swap Procedure

s	2	3	4	5	6	7	8
Number of swaps	1	1	0	0	0	0	0
Solution before swap	982,371	932,247	905,261	889,597	880,831	878,304	874,472
Solution after swap	972,590	924,858	905,261	889,597	880,831	878,304	874,472
Gap improvement	1.00%	0.79%	0%	0%	0%	0%	0%

3.8 Conclusions and Future Research

We propose a strategic heuristic approach to select the most cost-effective intermediate hubs for cross-docking activities. The selection of hubs adds another layer of complexity on top of service network design. To address this, we develop decomposition approaches to make the problem tractable and to obtain high-quality solutions. We also make practical assumptions that arise in practice, which help to reduce the size of the problem. We implement our methods in a real-world data set provided by our courier partner. We show that our methodology produces good-quality solutions in terms of the locations of the intermediate hubs selected and in terms of gap performance. All different greedy variants we develop outperform the hub selection model, with all time-feasible one-stop paths, by a 20% gap difference for $s = 6, 7$.

The best greedy variant solutions return gaps of approximately 15%, which suggests that further enhancements are needed to improve current solutions and find tighter lower bounds of the problem. In a few cases, the GRASP-Hub and the swap procedure produce slightly better solutions compared to the other variants; the improvement obtained is generally no more than 2% of gap.

As a future research avenue, investigating new heuristics for the problem would be very interesting given that the methodologies we develop are all mainly based on a greedy sequential scheme. Heuristics to explore may be other variants of IP-based models. Exact methodologies, such as column generation, which may deliver better lower bounds, would also be intriguing to explore.

Another aspect that is not taken into account in our current approach is the capacity, in terms of packages, that hubs permit for cross-docking. We currently assume it is unrestricted; however, this may turn into an issue for small hubs located in central geographical areas, in which high volume demand has to be managed.

CHAPTER 4
PATH GENERATION BASED HEURISTIC FOR SERVICE NETWORK DESIGN

4.1 Introduction

We consider a freight flow planning problem for a package courier system operating fast time-definite services. The setting in which we develop this work is identical to that considered in Chapter 3. In this work, we focus on a consolidation freight flow planning problem that conforms to a special structure that we term *generalized in-tree*, in which we assume that the terminals for intermediate sorting are selected or fixed. Since the network is specified, we aim to build a plan where path flows form a *generalized in-tree* for each destination, a concept that we introduce in detail in subsection 4.1.1. This structure intends to generalize the concept of an in-tree in which all flows at a terminal with a common destination must transfer next to the same hub. We extend this concept by incorporating a time requirement for the package into the definition, more specifically, the remaining available time for packages to reach their destinations without violating the service constraint. In other words, we expand the set of instructions for a terminal, compared to the in-tree structure, to include how packages are routed over the network depending on their destination and their remaining available time. In practice, this structure not only allows to enhance operational realism but also offers a seamless way to operate.

In this work, we consider the region we employ in Chapter 3, the southern area of China that is composed of 29 cities and 69 hubs. In this region, 4 million packages are handled on average every day, which is equivalent to 120,000 commodities associated with an origin, destination and time requirement. Similarly as in Chapter 3, the setting described imposes a scale challenge to be tackled, and therefore, the approaches we develop must scale to the size of this problem. To do so, we develop decomposition heuristic approaches in which we build a dynamic path generation heuristic that seeks to find multi-stop paths that enables to take advantage of consolidation opportunities.

4.1.1 Generalized In-Tree Structure

We introduce the concept of a *generalized in-tree* structure, referred to as GIT, which has a foundation from an operational perspective that aims to simplify the set of instructions at terminals for how to plan the flow of packages. Secondly, we aim to generalize the concept of an in-tree structure which enables simple terminal operations because it allows only the verification of the destination of a package to determine the proper outbound trailer for loading. This operational strategy is simple in terms of instructions, nonetheless, it restricts the flow planning flexibility to take advantage of consolidation opportunities, and therefore, to be able to reduce operational costs further. On the other hand, a complete relaxation of this structure is the more flexible strategy where flows can be routed unrestricted through different terminals regardless of their destinations, origins and remaining available times. This relaxed plan is more difficult to operate in practice, especially when commodities are not restricted to follow single all-or-nothing paths from origin to destination.

A more realistic and flexible approach than an in-tree structure, but one that still imposes some structure on the solution that enables practical benefits during operations is what we term a GIT structure. We say that paths conform to a GIT structure (i, d, [α, β), j) if all flow at terminal i with a final destination d and having a remaining available time between [α, β) will transfer next to terminal j. We call the interval [α, β) a *bucket* for a given terminal i and destination d. We show an example of this concept in Figure 4.1, where at terminal i, there are three buckets for two flows of packages passing through terminal i and with a common destination d. Since the remaining available time of flow k_1 falls into the first bucket, $[1, 3)$, the flow will transfer next directly to destination d. The remaining available time of flow k_2 falls into the last bucket, $[4, -)$ (which means any remaining available time greater than or equal to 4), then it will transfer next to intermediate hub B.

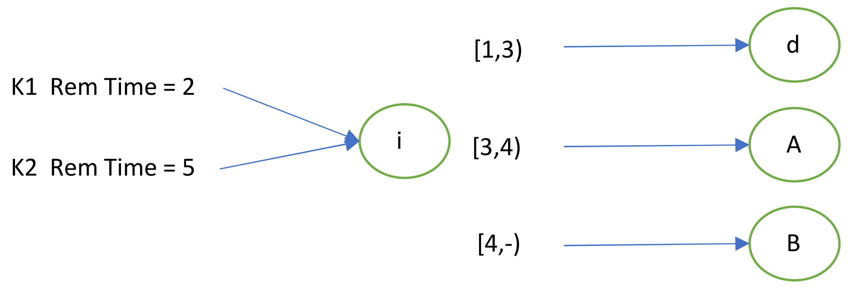

Figure 4.1: Illustration of a GIT Structure

The definition of a GIT structure is flexible enough to arbitrarily define many high-granularity buckets, i.e. when $\beta - \alpha$ is small for each bucket. Therefore, in general it is true that any set of flows will conform to an arbitrary GIT structure. The exceptional case occurs when there are two flows with the same remaining available time at terminal i, with a common destination d, which transfer to a different next terminal. This rarely occurs because we use rational distances and travel times on arcs. Figure 4.2 depicts an example of three flows, with different remaining available times at terminal i, which we convert into a GIT structure by defining suitable buckets constructed from the remaining available times of the flows at terminal i.

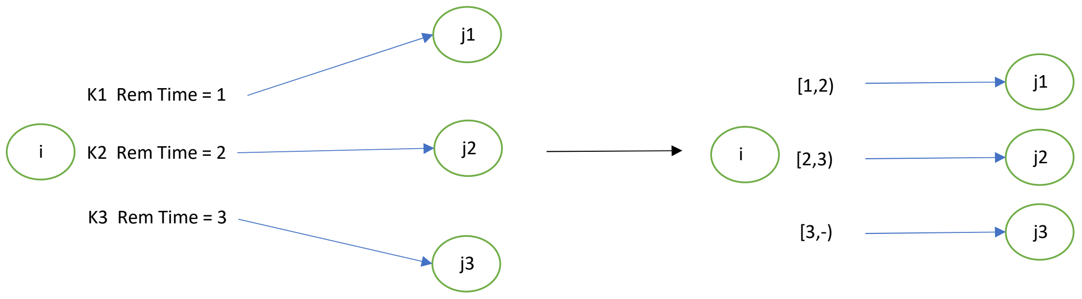

Figure 4.2: Construction of a GIT Structure

As we previously discussed, given the fact that (almost always) any set of flows form an arbitrary GIT structure, it is helpful to instead introduce a discretized version of the GIT structure. The discretized version adds on top of the definition of the GIT the following restrictions: we constrain all buckets [α, β) to be of a fixed width of B hours, i.e., α-$\beta = B$, except for the last bucket which is of the form [α, -); the first bucket starts from zero and they do not intersect, i.e. buckets are of the form [0,B), [$B, 2B$) and so on. Thus, when we refer to this structure, we term it as a B-hour discretized GIT. It is noteworthy to mention that the example in Figure 4.2 cannot be modified into a 2-hour discretized GIT, because there are two remaining available time flows that fall into the bucket [2,4) that transfer to two different next terminals.

4.1.2 Flow Planning with GIT Structure

The novelty of the work presented in this chapter is that we introduce the concept of a GIT and a discretized GIT, which allows the creation of structured flows plans that are more flexible but provide useful operational benefits. For flow planning under this structure, we develop and compare a number of different approaches: a GIT-based approach, an optimized GIT-based formulation and a relaxation using a path-based formulation. We develop also novel dynamic path generation procedures for the GIT-based and path-based formulations, because we cannot enumerate all time-feasible paths given the set sizes that would result on medium and large problems.

Given that realistically-sized problems can yield large integer programming formulations, we model our approaches on a flat network, where we consider flows of packages associated with an origin, destination and time requirement. We aim to develop dynamic path generation approaches that seek to find flow plans comprised of good multi-stop paths, where planned solutions conform to a reasonable discretized GIT structure for each destination. The path generation heuristic that we develop is inspired by the concepts of column generation where dual prices are used to compute arc reduced costs after solving specially-designed linear programs that capture some of the benefits of consolidation.

We develop three flow planning approaches in this chapter: (1) a relaxation of the problem using a path-based formulation, (2) a GIT-based approach, and (3) an optimized

GIT-based approach. We relate and compare results to Chapter 3. We demonstrate, via a computational study, that the path-based approach that enforces single paths for commodities generally leads to a 4-hour discretized GIT structure for small instances; for medium and large instances, the solutions do not conform to a discretized GIT structure overall, however, more than 90% of the solution does conform to a 2 and 4-hour discretized GIT structure. The flexibility to use multi-stop paths, having more than one intermediate stop, using the same set of intermediate hubs selected in Chapter 3, permits us to improve the flow planning solution costs from that chapter by more than 3% when using the cost structure of the large truck type. We show that imposing different GIT structures creates solution penalty costs between 2% and 4%. Computational experiments show that the three approaches behave comparatively similar for the smallest sized instances, nevertheless, the path-based and the optimized GIT-based approaches strongly outperform the GIT-based approach by more than 10% of gap performance for the large size set of instances. Interestingly, when imposing a tree structure on the optimized GIT-based approach, the gap performance difference is 3.44% compared to the solutions of the relaxation of the problem through the path-based approach for the largest instance.

The contributions of our work are summarized as follows:

- We generalize the in-tree concept by introducing the GIT structure which has useful operational benefits.
- We develop novel GIT-based and optimized GIT-based approaches.
- We build dynamic path generation approaches to avoid enumerating all time-feasible paths in advance.
- We demonstrate that different GIT structures yield a penalty cost between 2% and 4% for the data set used.
- We demonstrate that approaches that dynamically create multi-stop paths permit us

to improve solution costs by about 3% in comparison to the setting in chapter 3 in which we allow at most one-stop for paths.

The remainder of the work is structured as follows. In section 4.2, we present relevant prior literature review. In section 4.3, we provide a formal description of the problem. In section 4.4, we present the relaxation of the problem through a path-based formulation. In section 4.5, we describe the GIT-based and the optimized GIT-based approaches. In section 4.6, we conduct a computational study based on a real world data set that we employed in Chapter 3. Finally, in section 4.7, we present conclusions and future research.

4.2 Literature Review

Most of the main literature and works on service network design, hub locations and hub network design problems are presented in the literature review of Chapter 3. We add relevant literature and works related to column generation approaches and branch-and-price techniques on service network design and bi-objective shortest path problems that are related to the work we develop in this Chapter.

One of the first frameworks of column generation for solving general large problems is presented in [50]. [5] presents a branch-and-price-and-cut approach to solve origin-destination integer multicommodity flow problems. In this work, they develop an enhanced branch-and-price algorithm by adding cuts, and demonstrate that by generating columns only at the root node, the branching rule is more effective at finding the optimal solution than a standard rule. In [6], a branch-and-price algorithm is presented for addressing the service network design problem with asset management constraints (SNDAM), in which they integrate two column generation subproblems for integer cycle design and continuous flow-path variables of the problem. In [7], a branch-and-price-and-cut (BPC) algorithm is developed for the multicommodity capacitated fixed-charge network design problem, in which the restricted master problem is a compact arc-based model by considering only a subset of the commodity flow variables. The pricing subproblem corresponds to a lagrangian relaxation of the flow conservation and capacity constraints. The BPC performs well and is more efficient than an MIP solver and a branch-and-cut algorithm that does not integrate column generation. [8] also develops a branch-and-price approach for the directed network design problem with relays (DNDR), and proposes two formulations for this problem: a node-arc and an arc-path. For the latter, they formulate an arc-path formulation for which they develop a branch-and-price approach with two methods for entering columns. [51] addresses an integrated tactical planning service network design and hub location problem for road-rail transport, they develop a path-based model that is solved by branch-and-price-cut which outperforms a commercial solver employing an arc-based approach. In [52], they exploit a tree formulation along with a branch-and-price-and-cut algorithm for the single-source demand case of the network design problem with relays. We find column generation based algorithms for the location-routing problems in [53] and [54].

On algorithms for solving the bi-objective shortest path, one of the most common algorithms is the bound label setting algorithm (Blset) in [55], which is based on a label setting algorithm that employs prune techniques by using labels at nodes and labels at the final node which are referred to as bounds (also called efficient set). For the one-to-one case (single origin-destination), results are compared to the label correcting algorithm with and without bounds in which Blset is the best performer. A recent new method is the Pulse algorithm, in [56], which is based on a depth first search recursive algorithm truncated by pruning strategies. Basically it creates a complete solution to the destination, that they call ”pulse” which may be an efficient solution. To avoid large enumeration, it prunes along the way by nadir point (anti-ideal point), by cycles, efficient set and labels. It performs better than the Blset algorithm in almost all instances tested for the one-to-one case. For the one-to-all case, it may not be efficient because there are many nadir points as destinations and it needs to solve a single shortest path for each objective and destination.

A recent algorithm is the label setting with dynamic update of pareto front (LSDFP) in [57]. It is based on a label setting and has a first phase to find the non-dominate pareto points. The algorithm solves one objective shortest path and convex combinations for pruning, dynamically updates the whole pareto front, and develops 3 strategies for selecting labels: lexicographic, sum of costs, and convex combinations of costs. It performs better than the Blset and the Pulse algorithms for medium and large instances, but not for small instances.

One of the the latest algorithms, up to this date, is the bi-objective Dijkstra method (BDijkstra), in [58], which is based on a label setting algorithm. They develop a unidirectional and bidirectional variant and use a lexicographic label selection. For the one-to-one case, it adds pruning strategies, tests bidirectional scheme, and computes the nadir point (anti ideal point) for further pruning. Both variants of the algorithm, the unidirectional and bidirectional, perform better than both the Pulse and the Blset algorithms in the instances studied. In our work, we solve bi-objective shortest path problems for the path and GIT generation heuristics of the GIT-based and path-based formulations.

4.3 Problem Description

We consider an inter-city service network where packages need to be moved from their origins to their destinations. The network is defined on a graph $\mathcal{G} = (\mathcal{U}, \mathcal{E})$ with node set $\mathcal{U}$ representing hubs and an arc set $\mathcal{E}$ representing (directed) transportation connections between terminals. We assume a distance d_e and a cost c_e on arc e corresponding to a vehicle movement of capacity Q_e. Let $\bar{\mathcal{U}}$ denote the set of hubs that are admissible as intermediate hubs, and it takes a time t_u^{cross} for $u \in \bar{\mathcal{U}}$. We assume unrestricted capacity at each hub $u \in \mathcal{U}$.

We define the time requirement of a package as the available time to reach its destination starting from its origin terminal. We define a commodity k to be the set of shipments sharing the same origin, destination and time requirement. Let $\mathcal{K}$ be the set of all commodities, where a commodity $k \in \mathcal{K}$ is identified by a tuple $(o_k, d_k, \delta_k, q_k)$, where $o_k \in \mathcal{U}$

is the origin, $d_k \in \mathcal{U}$ is the destination, δ_k is the time requirement and q_k is the commodity size measured as number of pieces.

The objective is to find time-feasible flow paths that conform to a GIT structure for each destination so as to minimize package movement costs on arcs while ensuring that all commodities respect their time requirements. Packages are sorted and reloaded at each intermediate terminal along the flow path from origin to destination.

4.4 Path-Based Formulation

This section focuses on solving a service network design problem with commodity service time constraints on a *flat network* using a formulation where the decision variables are paths, in which we define a path p as a sequence of terminals (or a sequence of arcs) $(u_1 = o_k, u_2, ..., u_k = d_k)$ that commodity k takes from its origin, o_k, to its destination, d_k, where each (u_i, u_j) is an arc in $\mathcal{E}$.

A potential commodity path p is said to be time-feasible if the sum of the travel times of all arcs of the path and the handling times (unloading and loading) at all of its intermediate sorting stops along the path is no greater than the time requirement, δ_k. That is,

$$\sum_{e \in p} t_e + \sum_{u \in p} t_u \leq \delta_k \tag{4.1}$$

4.4.1 Formulation

We present the path-based formulation on a flat network. The model selects one path per commodity in order to minimize the vehicle resources needed to transport the packages such that all commodities respect their time requirements. Such a path-based formulation is known as a single-path formulation.

Let $\mathcal{P}^k$ represent a set of time-feasible paths for commodity $k \in \mathcal{K}$ and let parameter γ_e^p equal 1 if path p covers arc e, and 0 otherwise. In addition to the set and parameters introduced before, we define the variables to be employed in the formulation. Let binary variable x_p^k equal 1 if path p is selected for commodity k, and 0 otherwise. Let integer variable y_e be the number of vehicles used on arc e. The model of selecting one time-feasible path per commodity such that package movement costs are minimized is as follows.

$$\min \quad \sum_{e\in\mathcal{E}} y_e c_e \tag{4.2}$$

$$\text{s.t} \quad \sum_{k\in\mathcal{K}}\sum_{p\in\mathcal{P}^k} \gamma_e^p q_k x_p^k \leq Q_e \cdot y_e \quad \forall e\in\mathcal{E} \tag{4.3}$$

$$\sum_{p\in\mathcal{P}^k} x_p^k = 1 \quad \forall k\in\mathcal{K} \tag{4.4}$$

$$x_p^k \in \{0,1\} \quad \forall k\in\mathcal{K}, \forall p\in\mathcal{P}^k \tag{4.5}$$

$$y_e \in \mathcal{Z}^+ \quad \forall e\in\mathcal{E} \tag{4.6}$$

In 4.2, we minimize the cost of arcs. In constraints 4.3, the commodity paths selected induce the number of trucks needed on each arc. Restrictions 4.4 select one path per commodity and constraints 4.5 and 4.6 are the domain variables.

4.4.2 Minimum Reduced Cost Path

We introduce the pricing problem to create a minimum reduced cost path for a commodity k. Let α_e and θ_k be the dual pricing values associated with the constraints 4.3 and 4.4, respectively. The reduced cost of the x_p^k variable to generate a path p for commodity k is computed as follows:

$$\tilde{c}_p^k = 0 - (\sum_{e\in\mathcal{E}} \gamma_e^p q_k \alpha_e + \theta_k) \tag{4.7}$$

Thus, for a commodity k, we aim to create a time-feasible path with a minimum reduced cost; when the reduced cost is negative for such a path, it may improve the objective function value of linear programming relaxation of the path-based formulation. Since we can take q_k and θ_k out of the sum, we therefore minimize:

$$\sum_{e \in \mathcal{E}} \gamma_e^p \alpha_e \tag{4.8}$$

Given the time requirement restriction of each commodity, the problem becomes solving a constrained shortest path problem for each commodity $k \in \mathcal{K}$. Instead of solving a constrained shortest path for each commodity, we solve a bi-objective shortest path problem for each destination $d \in \mathcal{D}$ to find all non-dominated paths, associated with the non-dominated points of the bi-objective shortest path solution, from all origins to destination d. We then identify the constrained shortest path for each commodity $k \in \mathcal{K}^d$ and $d \in \mathcal{D}$ depending on its time requirement.

The bi-objective shortest path algorithm is based on a backward label setting algorithm which starts from the destination d, and finds all non-dominated paths for each origin. Let δ_d^{max} be the maximum time requirement among all commodities with a common destination d, i.e. belonging to the set $\mathcal{K}^d$. We outline the bi-objective shortest path procedure in algorithm 3.

Algorithm 3 Bi-Objective Shortest Path

```
1: Set $L_d = \{(0,0)\}$, set of labels, and $L_i = \emptyset \ \forall i \in \mathcal{U} \setminus d$
2: $I_i$ index set of labels for each $i \in \mathcal{U}$ , set $I_d = (0,0)$
3: Set $T_i = \emptyset$ for each $i \in \mathcal{U}$
4: while $\bigcup_{j \in \mathcal{U}} I_j \setminus T_j \neq \emptyset$ do
5:     Choose minimum lexicographic label $l$, let $j$ be the associated node and $t_j^l$ and $c_j^l$ travel time and cost, respectively.
6:     Treat label $l$
7:     if $j \in \bar{\mathcal{U}}$ or $j = d$ then
8:         for $i \in \delta^-(j)$ do
9:             We say that label $l$ is dominated by label $m$ if $t_i^m \leq t_j^l + t_{ij}$ and $c_i^m \leq c_j^l + c_{ij}$
10:            if label $l$ is not dominated by any other label $m \in I_i$ and $t_j^l + t_{ij} \leq \delta_d^{max}$ then
11:                Delete labels dominated by label $l$
12:                Set $L_i = L_i \bigcup l$
13:                Add label $l$ to $I_i$
14:                Update $t, c, stops$ with the label $l$ at node $i$
15:            end if
16:        end for
17:        $T_i = T_i \bigcup l$
18:    end if
19: end while
```

Algorithm 3 works as follows. We first initialize the set of labels L at the destination d with cost and travel time zero and empty for all other nodes. We save the set of index labels in set I and the set of visited labels in set T for each node (from line 1 to 3). While we have unexplored labels, we select the minimum lexicographic label l associated with node j. Next, we verify whether the label contains a path with intermediate hubs in the admissible set $\bar{\mathcal{U}}$ and is not dominated by any other label at node j (from line 5 to 10). If so, we add the label l to the set of labels L, we delete the labels dominated by l, and we update the sets L, T and I, and parameters t, c and $stops$ (from line 11 to 17). We iterate until all labels are visited.

4.4.3 Path Generation Based Heuristic

In this part, we focus on developing a dynamic path generation heuristic procedure for the path-based formulation. In subsection 4.4.2, we present the pricing problem to generate a minimum reduced cost path for a commodity. Solving the LP relaxation of the path-based formulation offers no incentive for consolidating commodities because the optimal LP solution is simply to route commodities either over the direct or the shortest path. Since we aim to generate non-trivial multi-stop paths that take advantage of consolidation opportunities, we impose lower bounds on truck variables, y_e, because this way the optimal LP path flow is incentivized to use non-trivial paths containing arcs with the lower bounds. These lower bounds are driven by the solution from the IP of the path-based model. We construct the initial set of paths for all commodities using the direct routes. Thus, we aim to set lower bounds on truck variables that depend on the package flow of the IP solution, i.e., $f_e = \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}^k} \gamma_e^p q_k x_k^p$ for each $e \in \mathcal{E}$. To determine whether to set a lower bound on the truck variable on arc e, we randomly set a lower bound with probability p, i.e., if a uniform random number is lower or equal to p, we set a lower bound on arc e as $\lceil \frac{f_e}{Q_e} \rceil$, otherwise we do not set a lower bound. We round up the value of $\frac{f_e}{Q_e}$ because this is an effective lower bound for the LP relaxation when seeking integer solutions.

Given a set of imposed lower bounds, we solve the LP relaxation of the path-based model. Using the dual prices from this solution, we then solve a bi-objective shortest path problem for each destination to obtain optimal flow paths from all origins to d. All paths from origins to destination d correspond to minimum reduced cost paths. Depending on the time requirement of each commodity, we obtain the minimum reduced cost path for each commodity. If the reduced cost of the path is negative, we add the new generated path to the set of time-feasible paths. Then, we solve the IP of the path-based model anew with the path variables created; we note at this point, new lower bounds are generated for the linear program. We iterate in this fashion until a maximum number of n^{max} iterations is reached or the objective value does not vary more than $r\%$ after T iterations. We detail the procedure in algorithm 4.

Algorithm 4 Path Generation

```
1: Set $n = 1$ and $Stop = False$
2: Solve LP relaxation problem using the direct paths
3: Solve IP problem using the direct paths
4: Set lower bound $y_e^{min} = \lceil \frac{f_e}{Q_e} \rceil$ on arc $e$ with probability $p$
5: while $n \leq n^{max}$ and not $Stop$ do
6:     for $d \in \mathcal{D}$ do
7:         Solve a bi-objective shortest path problem for destination $d$
8:         for $k \in \mathcal{K}^d$ do
9:             Map the path associated with the origin $o_k$ and time requirement $\delta_k$ from
   the solution of the bi-objective shortest path problem, call it $p^k$
10:            Compute reduced cost of path $p^k$ as $\tilde{c}_p^k = -\theta_k - \sum_{e \in p^k} q_k \alpha_e$
11:            Add the path if reduced cost is negative
12:        end for
13:    end for
14:    Solve LP relaxation problem with path variables added
15:    Solve IP problem with path variables added
16:    Set lower bound $y_e^{min} = \lceil \frac{f_e}{Q_e} \rceil$ on arc $e$ with probability $p$
17:    $n \leftarrow n + 1$
18:    If objective value does not change more than $r\%$ in the last $T$ iterations set $Stop =$
   $True$
19: end while
```

Algorithm 4 works as follows. First, we initialize parameters of iterations (line 1). We solve initial LP and IP of the path-based formulation by using the direct paths for all commodities. Next, we set lower bounds on truck variables, y_e, for each arc $e \in \mathcal{E}$ depending on probability p (line 4). In each iteration, we employ the dual pricing values from the LP relaxation solution to solve a bi-objective shortest path problem for each destination $d \in \mathcal{D}$.

We thus obtain the constrained shortest path for each commodity $k \in \mathcal{K}^d$ in order to compute the reduced cost of the path (lines 9 and 10). We add the constrained shortest path for each commodity if the reduced cost is negative (line 11). We solve anew the LP relaxation and IP problems with the new generated paths (lines 14 and 15). We set lower bound on truck variables using the new IP problem solved (line 16). We iterate in this manner until a maximum number of n^{max} iterations is reached or the objective value does not change more than $r\%$ after T iterations.

4.5 GIT-Based Formulation

This section focuses on solving a service network design problem with commodity service time constraints on a *flat network* by a formulation where the decision variables are GITs defined for destinations $d \in \mathcal{D}$, where $\mathcal{D}$ is defined as the set of destinations such that at least one commodity $k \in \mathcal{K}$ has a destination $d(k) = d$.

To understand the GIT-based formulation, suppose that a solution selects a GIT G_g^d for each destination d. For each commodity $k \in K$ where $d(k) = d$, the selected GIT G_g^d defines a unique path from origin to destination for that commodity. Thus, the selection of G_g^d for destination d implies freight volumes $f_g^{e,d}$ that need to be added to each truck dispatch arc $e \in \mathcal{E}$. Let $\mathcal{G}^d$ represent a set of time-feasible GITs for destination $d \in \mathcal{D}$. Thus, a constraint to determine the number of trucks needed on arc $e \in \mathcal{E}$ is as follows:

$$\sum_{d \in D} \sum_{g \in \mathcal{G}^d} f_g^{e,d} x_g^d \leq \tau_e \tag{4.9}$$

where the variable x_g^d is binary and equal to one if GIT g is selected for destination d in the solution, and 0 otherwise.

4.5.1 Formulation

We present the GIT-based formulation on a flat network. The model selects one GIT per destination that aims to minimize the vehicle resources needed to transport the packages such that all commodities meet their corresponding time requirements.

We say that a GIT for a destination d is time-feasible if the unique paths from their origins to the destination are time-feasible for all commodities, i.e., the travel time of the path is less than or equal to δ_k for each commodity k with a common destination d.

In addition to the set and parameters introduced before, we define the variables to be employed in the formulation. Let binary variable x_g^d equal 1 if GIT g is selected for destination d, and 0 otherwise. Let integer variable y_e be the number of vehicles used on arc e. The model of selecting one time-feasible GIT per destination such that package movement costs are minimized is as follows.

$$\begin{aligned}
\min \quad & \sum_{e \in \mathcal{E}} y_e c_e & & \qquad (4.10)\\
\text{s.t} \quad & \sum_{d \in \mathcal{D}} \sum_{g \in \mathcal{G}^d} f_g^{e,d} x_g^d \le Q_e \cdot y_e & \forall e \in \mathcal{E} & \qquad (4.11)\\
& \sum_{g \in \mathcal{G}^d} x_g^d = 1 & \forall d \in \mathcal{D} & \qquad (4.12)\\
& x_g^d \in \{0,1\} & \forall d \in \mathcal{D}, \forall g \in \mathcal{G}^d & \qquad (4.13)\\
& y_e \in \mathcal{Z}^+ & \forall e \in \mathcal{E} & \qquad (4.14)
\end{aligned}$$

In 4.10, we minimize the cost of arcs. Constraints 4.11 determine the number of trucks needed on each arc induced by the GITs chosen. Restrictions 4.12 select one time-feasible GIT per destination, and constraints 4.13 and 4.14 are the domain variables.

4.5.2 Minimum Reduced Cost GIT

We outline the pricing problem to create a GIT for destination d. Let α_e and θ_d be the dual pricing values associated with the constraints 4.11 and 4.12, respectively. The reduced cost of the x_g^d variable to generate a GIT g for destination d is computed as follows:

$$\tilde{c}_g^d = 0 - (\sum_{e \in \mathcal{E}} f_g^{e,d} \alpha_e + \theta_d) \tag{4.15}$$

Thus, for a destination d, we aim to create a GIT with a minimum negative reduced cost.

We solve the problem of creating a minimum negative reduced cost GIT by finding the constrained shortest paths from all commodity origins to destination d. In turn, this problem can be decomposed by commodity with a common destination d. Let $\mathcal{K}^d$ denote the set of commodities such that $d(k) = d$.

Let us suppose the solution of the constrained shortest path for commodity $k \in \mathcal{K}^d$ is p^k. Then, the reduced cost of a GIT for destination d, and the associated set of commodities $\mathcal{K}^d$, can be written as follows:

$$\tilde{c}_g^d = -\theta_d - (\sum_{k \in \mathcal{K}^d} \sum_{e \in p^k} q_k \alpha_e) \tag{4.16}$$

Since we aim to solve all constrained shortest paths from all commodity origins to a destination d, instead of solving a constrained shortest path for each commodity, we solve a bi-objective shortest path problem for a destination d to find all non-dominated paths, associated with the non-dominated points of the bi-objective shortest path solution, from each origin to destination d. Then, we identify the constrained shortest path for each commodity depending on its time requirement.

Bi-Objective Shortest Path Problem

We solve a bi-objective shortest path problem for each destination which has been extensively studied in recent works for the one-to-one and the one-to-many cases (see [55], [56], [57] and [58]). For a given destination d, we find all non-dominated points for all origins. At an origin i, the non-dominated points are of the form (γ, j), which indicates the shortest path, from i to d, by transferring next to j when the remaining available time is greater or equal to γ. We employ these non-dominated points to build a GIT and then modify it to obtain a discretized GIT structure for each destination.

Construction of a Discretized GIT Structure

We detail how we build a discretized GIT structure from the solution of the bi-objective shortest path problem. Let us suppose we have N non-dominated points (associated to shortest paths) at terminal i for which we know the travel time t_l, from i to d, and the next terminal to transfer to, j_l, for each $l = 1, ..., N$. We sort the times t_l from shortest to longest and we construct the buckets at terminal i in the form of (i, d, [t_l, t_{l+1}), j_{l+1}) for each $l = 1, ..., N - 1$ and a last bucket (i, d, [t_N, -),j_N). These buckets specify what the next terminal to transfer to is depending on what bucket the remaining available time of a commodity falls into at terminal i. In Figure 4.3, we depict an example with $N = 4$ non-dominated points at terminal i, which we build into buckets.

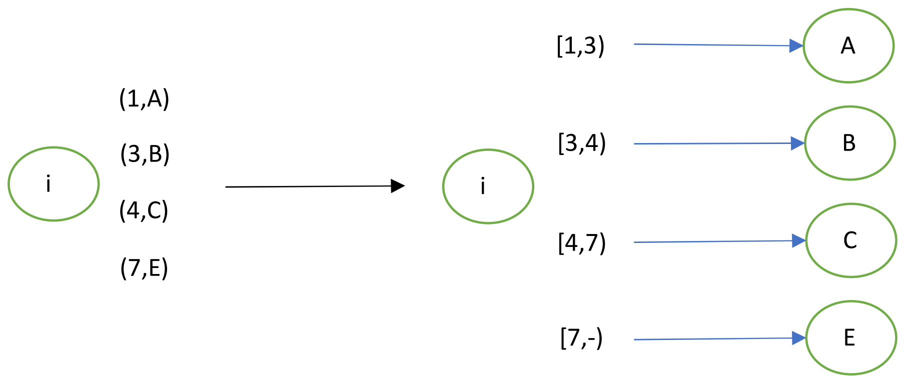

Figure 4.3: Construction of Buckets for a GIT Structure

The definition of a B-hour discretized GIT requires only one next transferring terminal for each bucket. Since the solution from the bi-objective shortest path does not necessarily guarantee to respect this restriction for all discretized buckets, we then modify the GIT we construct in order to build a B-hour discretized GIT. To do so, we need to ensure time feasibility of commodity paths. This means that the next terminal j to transfer to at terminal i for a given discretized bucket, cannot be a terminal for which it does not define a time-feasible path, from j to d, for any commodity passing through or originating at terminal i that falls into a given discretized bucket.

We modify the GIT to obtain a B-hour discretized GIT in the following manner; we define the next terminal j to transfer to at terminal i for a discretized bucket as the next terminal of the buckets, from GIT, intersecting the discretized bucket with the smallest interval bounds. This way, selecting the next terminal to transfer to from the smallest interval bound bucket with the discretized bucket guarantees that the modified paths are time-feasible for any remaining available time of a commodity at terminal i. The latter is explained by the fact that the paths, from the GIT, defined by the buckets built from the non-dominated points of the bi-objective shortest path solution, are already time-feasible, therefore, when we map the next terminal to transfer to, associated with an equal or faster travel time path, to a discretized bucket, it still remains time-feasible. On the other hand, we implicitly enforce commodities with a greater remaining available time within a given discretized bucket to possibly be routed over a faster but a more expensive path. Depending on the width of the discretized bucket, B, we may end up routing commodities over shorter routes for large values of B, and over more cost-efficient paths for small values of B. Formally, let us suppose we have a GIT in which at terminal i is composed of N ordered buckets [α_i,β_i) for each $i = 1, ..., N$, (constructed from the non-dominated points of the solution of the bi-objective shortest path). Thus, we select the next transferring terminal j_{i^*} for a discretized bucket [α,β) as follows.

$$i^* = \min_{i \in \{1,...N\}} \{i \in \{1,...N\} | [\alpha_i, \beta_i) \cap [\alpha, \beta) \neq \emptyset\} \tag{4.17}$$

Figure 4.4 illustrates the same example as Figure 4.3 modified to a 2-hour discretized GIT structure. Discretized bucket [2,4) intersects with buckets [1,3) and [3,4), then given that the bucket [1,3) is the smallest interval bound, we set terminal A as the next transferring terminal of discretized bucket [2,4) and we similarly set the next transferring terminal for all the other discretized buckets.

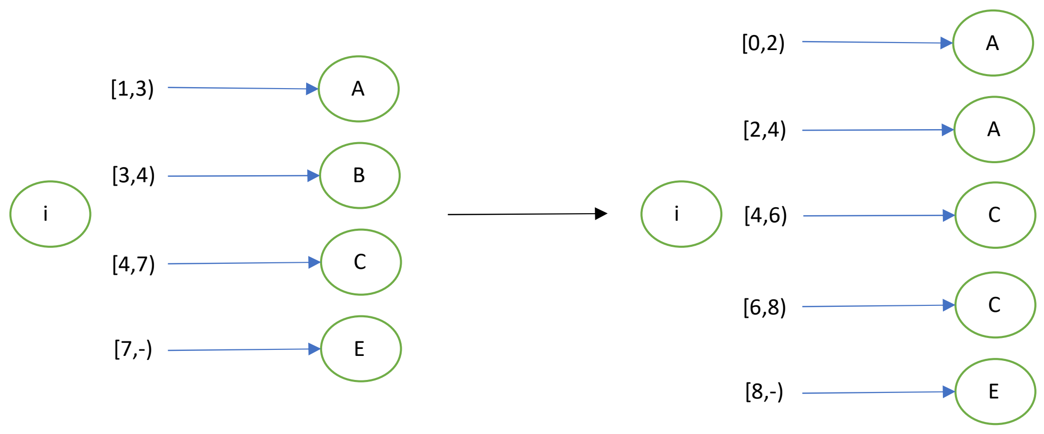

Figure 4.4: Modification to a Discretized GIT Structure

4.5.3 GIT Generation Based Heuristic

In this part, we focus on developing a GIT generation heuristic procedure for the GIT-based formulation. The core procedure is very similar to the path generation heuristic for the path-based formulation presented in subsection 4.4.3, although, here we generate GITs. In subsection 4.5.2, we presented the pricing problem to create a GIT and we detail how to modify a GIT to obatain a discretized GIT. The main idea is to solve an iterative scheme, in which at each iteration, we solve the LP relaxation of the GIT-based formulation with lower bounds on a subset of truck arcs. We impose lower bounds on truck variables, y_e, of the LP relaxation of the GIT-based model in order to generate non-trivial multi-stop paths. These lower bounds are driven by the solution from the IP of the GIT-based formulation. We construct the initial discretized GIT for each destination induced by the direct paths. First, we aim to set lower bounds on truck variables that depend on the package flow of the integer programming (IP) solution, i.e., $f_e = \sum_{d \in \mathcal{D}} \sum_{g \in \mathcal{G}^d} f_g^{e,d} x_g^d$ for each $e \in \mathcal{E}$. To determine whether to set a lower bound on the truck variable on arc e, we randomly set a lower bound with probability p, i.e., if a random number is lower or equal to p, we set the lower bound on arc e as $\lceil \frac{f_e}{Q_e} \rceil$, otherwise we do not set a lower bound.

Then, we solve the LP relaxation of the GIT-based model imposing the lower bounds set. Using the dual pricing values from the LP relaxation solution, we solve a bi-objective shortest path problem for each destination, and from this solution, we create a minimum reduced cost GIT, which we modify to a B-hour discretized GIT. Next, we add a new generated B-hour discretized GIT for a destination if the reduced cost is negative. We solve the IP of the GIT-based model anew with the B-hour discretized GIT variables generated. We iterate in this fashion until a maximum number of n^{max} iterations is reached or the objective value does not vary more than $r\%$ after T iterations. We detail the procedure in algorithm 5.

Algorithm 5 GIT Generation

```
1: Set $n = 1$ and $Stop = False$
2: Solve LP relaxation model with initial B-hour discretized GIT variables induced by direct paths
3: Solve IP model with initial B-hour discretized GIT variables induced by direct paths
4: Set lower bound $y_e^{min} = \lceil \frac{f_e}{Q_e} \rceil$ on arc $e$ with probability $p$
5: while $n \leq n^{max}$ and not $Stop$ do
6:     for $d \in \mathcal{D}$ do
7:         Solve a bi-objective shortest path problem for destination $d$
8:         Build the $B$-hour discretized GIT for each destination $d$
9:         For each commodity $k \in \mathcal{K}^d$, map the path associated with the origin $o_k$ and time requirement $\delta_k$ from the $B$-hour discretized GIT, call it $p^k$
10:        Compute reduced cost of the $B$-hour discretized GIT for destination $d$ as $\tilde{c}^d = -\theta_d - (\sum_{k \in \mathcal{K}^d} \sum_{e \in p^k} q_k \alpha_e)$
11:        Add $B$-hour discretized GIT if the reduced cost is negative
12:    end for
13:    Solve LP relaxation model with discretized GIT variables added
14:    Solve IP model with discretized GIT variables added
15:    Set lower bound $y_e^{min} = \lceil \frac{f_e}{Q_e} \rceil$ on arc $e$ with probability $p$
16:    $n \leftarrow n + 1$
17:    If objective value does not change more than $r\%$ in the last $T$ iterations set $Stop = True$
18: end while
```

Algorithm 5 works as follows. First, we initialize parameters of iterations (lines 1 and 2). We solve the initial LP and IP of the GIT-based formulation using the discretized GIT variables induced by the direct paths for all commodities. Next, we set lower bounds on truck variables, y_e, for each arc $e \in \mathcal{E}$, depending on probability p (line 4). In each iteration, we employ the dual pricing values from the LP relaxation of the GIT-based solution to solve a bi-objective shortest path problem for each destination. We then construct a B-hour discretized GIT for each destination and we map the path, according to its time requirement, for each commodity $k \in \mathcal{K}^d$ in order to compute the reduced cost of the discretized GIT (line 9 and 10). We add the discretized GIT if the reduced cost is negative (line 11). We solve anew the LP relaxation and IP of the GIT-based model with the new generated discretized GIT variables (line 13 and 14). We set the lower bound on truck variables using the new IP problem solved (line 15). We iterate in this manner until a maximum number of n^{max} iterations is reached or the objective value does not change more than $r\%$ after T iterations.

4.5.4 Optimized GIT-Based Formulation

We present a second formulation of the GIT-based model that makes use of a pool of pre-generated time-feasible paths for commodities to build an optimized version of the discretized GIT structure. The model minimizes the vehicle resources needed to transport the packages such that a B-hour discretized GIT is built for each destination.

We define the following parameters and sets for the formulation. Let $\mathcal{E}^d$ be the set of tuples in the form of (i, j, b), which indicates that there is at least a commodity path, from the pool of paths, passing through or originating at terminal i (not the destination) having a next terminal j with a remaining available time within bucket b at terminal i, and let parameter $\Gamma^{p,k}_{i,j,b}$ be equal to 1 if this occurs, and 0 otherwise. Likewise, let $\mathcal{U}^d$ represent the set of tuples in the form of (i, b) which denotes that there is at least a commodity path passing through or originating at terminal i with a remaining available time within bucket b at terminal i. In addition to the set and parameters introduced before, we define the variables to be employed in the formulation. Let binary variable x^p_k equal 1 if path p is selected for commodity k, and 0 otherwise. Let integer variable y_e be the number of vehicles used on arc e. Let binary variable $z^d_{i,j,b}$ equal 1 if at terminal i there is at least one selected commodity path with a remaining available time within bucket b and heading to terminal j, and 0 otherwise. The model of selecting one time-feasible path per commodity such that each destination conforms a B-hour discretized GIT while package movement costs are minimized is as follows.

$$\min \quad \sum_{e \in \mathcal{E}} y_e c_e \tag{4.18}$$

$$\text{s.t} \quad \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}^k} \gamma_e^p q_k x_p^k \leq Q_e \cdot y_e \qquad \forall e \in \mathcal{E} \tag{4.19}$$

$$\sum_{k \in \mathcal{K}^d} \sum_{p \in \mathcal{P}^k} \Gamma_{i,j,b}^{p,k} x_p^k \leq z_{i,j,b}^d |\mathcal{K}^d| \qquad \forall d \in \mathcal{D}, \forall (i,j,b) \in \mathcal{E}^d \tag{4.20}$$

$$\sum_{(i,j,b) \in \mathcal{E}^d} z_{i,j,b}^d \leq 1 \qquad \forall d \in \mathcal{D}, \forall (i,b) \in \mathcal{U}^d \tag{4.21}$$

$$\sum_{p \in \mathcal{P}^k} x_p^k = 1 \qquad \forall k \in \mathcal{K} \tag{4.22}$$

$$x_p^k \in \{0,1\} \qquad \forall k \in \mathcal{K}, \forall p \in \mathcal{P}^k \tag{4.23}$$

$$z_{i,j,b}^d \in \{0,1\} \qquad \forall d \in \mathcal{D}, \forall (i,j,b) \in \mathcal{E}^d \tag{4.24}$$

$$y_e \in \mathcal{Z}^+ \qquad \forall e \in \mathcal{E} \tag{4.25}$$

In 4.18, we minimize the cost of selected arcs. In constraints 4.19, commodity paths induce the number of trucks needed on each arc. Restrictions 4.20 activate variable $z_{i,j,b}^d$ if there is at least one selected path passing through or originating at terminal i having a next terminal j with a remaining available time within bucket b for destination d. Constraints 4.21 limit, at terminal i, at most one next terminal j for each bucket b and destination d so as to form a B-hour discretized GIT. Finally, constraints 4.22 select one path per commodity and restrictions 4.23, 4.24 and 4.25 are the domain variables.

4.6 Computational Study

We test our approaches presented in the preceding sections by conducting a computational study based on real-world data provided by our Chinese courier company partner. For the experiments, we work with the way-bill of inter-city and intra-city packages for April 2019. From this data, we compute an average weight per day and an average number of parcels per day for each commodity.

The network is composed of 69 hubs. Taking into account all possible origin-destination pairs, we may have up to 4692 commodities without considering time requirements. We use the same base instance constructed in Chapter 3, i.e., we group time requirements of commodities according to the different groups of time requirements described in Chapter 3.

From the original instance that is composed of 69 hubs and 5297 commodities, which we refer to as the All-hubs instance, we build three groups of instances: small, medium and large size instances. In doing so, to create an instance, we select a given number of hubs, which we select in two variants; a random selection and geographically disperse selection. For each instance we create, its network is composed of all the arcs associated with the hubs selected. The commodities for the instance are defined as the commodities from the original instance associated with the hubs chosen, i.e., all commodities with an origin and a destination within the set of hubs selected. We detail this set of instances in subsection 4.6.2. For the GIT-based and the optimized GIT-based approaches, we build a 2-hour discretized GIT structure for most of the experiments we present.

Given the random nature of the setting of the lower bounds on the arc variables of the path and GIT-based generation heuristic approaches, each instance is run 10 times, and thus we report the best solution found.

Initial fine tuning tests in the All-hubs instance suggest that setting a probability of $\frac{1}{2}$ to determine lower bounds leads to the best gap performance results, in terms of objective value, on average. Therefore, we use this probability value for all the experiments we present.

4.6.1 Metrics

The purpose of this subsection is to define metrics that allow us to compare solutions across different experiments. We define the following metrics that permit us to verify whether solutions form a discretized GIT structure and how effectively resources are utilized.

- GIT B-hour: This metric verifies whether solutions form a B-hour discretized GIT, referred to as GIT BH. To check the structure, the metric verifies, for a given destination d, terminal i and each non-intersecting bucket $[0, BH), [BH, 2BH), ..., [(B^{max} - 1)H, B^{max}H)$, if all remaining available times of commodities passing through or originating at terminal i have the same next terminal (B^{max} is the minimum value such that $B^{max}H > \delta_i^{max}$, and δ_i^{max} is the maximum time requirement among commodities at terminal i). If there is at least one bucket having 2 different next terminals, then we say the pair (i, d) does not respect the B-hour discretized GIT structure for terminal i and destination d. Let $\mathcal{L}$ denote the set of pairs (i, d) for which there exists a commodity path passing through or originating at terminal i with destination d, and let parameter indicator $I_{i,d}$ be equal to 1 if the pair (i, d) complies with a B-hour discretized GIT structure, and 0 otherwise. The metric is computed as the percentage of all (i, d) pairs respecting the B-hour discretized GIT structure over all pairs in set $\mathcal{L}$, as follows:

$$\text{GIT B-hour} = \frac{\sum\limits_{(i,d)\in\mathcal{L}} I_{i,d}}{|\mathcal{L}|} \tag{4.26}$$

- Weighted GIT B-hour : We define the weighted B-hour discretized GIT metric that verifies whether solutions form a B-hour discretized GIT structure in which each (i, d) pair is weighted by the commodity volume passing through or originating at terminal i with destination d. Let $v_{i,d}$ be the sum of all commodity volume passing through or originating at i with destination d, then the metric is computed as follows:

$$\text{Weighted GIT B-hour} = \frac{\sum\limits_{(i,d)\in\mathcal{L}} I_{i,d}v_{,d}}{\sum\limits_{(i,d)\in\mathcal{L}} v_{i,d}} \tag{4.27}$$

- **Arc utilization:** This metric aims to capture how effectively the truck resources selected are utilized. We define the arc utilization metric as the sum of all commodity volume flows over each arc $e \in \mathcal{E}$, denoted by f_e, divided by the total capacity of trucks allocated over the network. Let y_e be the number of trucks selected on arc e, then the metric is computed as follows:

$$\text{Arc utilization} = \frac{\sum\limits_{e\in\mathcal{E}} f_e}{\sum\limits_{e\in\mathcal{E}} Q_e y_e} \tag{4.28}$$

4.6.2 Instances

For the small size set of instances, we generate 8 instances; 4 of 10 hubs where hubs are randomly selected and 4 of 10 hubs where hubs are selected geographically dispersed. We refer to these instances as 10-hub Rand1, 10-hub Rand2, 10-hub Rand3 and 10-hub Rand4 for the random selection variant, respectively, and 10-hub Geo1, 10-hub Geo2, 10-hub Geo3 and 10-hub Geo4 for the geographical selection variant, respectively. In Table 4.1, we show the number of commodities and destinations of the small size set of instances.

Table 4.1: Description of the Small Size Set of Instances

	Nº of commodities	Nº of destinations
10-hub Rand1	120	10
10-hub Rand2	125	9
10-hub Rand3	170	9
10-hub Rand4	65	7
10-hub Geo1	51	8
10-hub Geo2	52	9
10-hub Geo3	91	8
10-hub Geo4	94	10

For the medium size set of instances, we create 4 instances; 20 hubs where hubs are randomly selected, 20 hubs where hubs are chosen geographically dispersed, 30 hubs where hubs are randomly selected and 30 hubs where hubs are chosen geographically dispersed. We refer to these instances as 20-hub Rand, 20-hub Geo, 30-hub Rand and 30-hub Geo, respectively. In Table 4.2, we report the number of commodities and destinations of the medium size set of instances.

Table 4.2: Description of the Medium Size Set of Instances

	Nº of commodities	Nº of destinations
20-hub Rand	400	17
20-hub Geo	191	19
30-hub Rand	1133	28
30-hub Geo	891	29

Similarly, for the large size set of instances, we create 4 instances; 40 hubs where hubs are randomly selected, 40 hubs where hubs are chosen geographically dispersed, 50 hubs where hubs are randomly selected and 50 hubs where hubs are chosen geographically dispersed. We refer to these instances as 40-hub Rand, 40-hub Geo, 50-hub Rand and 50-hub Geo, respectively. We also include the original instance which we refer to as the All-hubs instance. Table 4.3 shows the number of commodities and destinations of the large size set of instances.

Table 4.3: Description of the Large Size Set of Instances

	Nº of commodities	Nº of destinations
40-hub Rand	2315	38
40-hub Geo	1634	38
50-hub Rand	3238	45
50-hub Geo	3138	47
All-hubs	5297	58

4.6.3 Primary Results

In this set of experiments, we aim to compare the flow planning results from Chapter 3 when using paths with at most one stop to the path-based and the optimized GIT-based approaches when no restriction is imposed on the number of intermediate sorts in a path. For this comparison, we will use the set of intermediate sorting hubs selected by the approach in Chapter 3 and simply examine the flow planning results given those hubs. To do so, we run the hub selection approach from Chapter 3 to obtain the 6, 7 and 8 most cost-efficient intermediate hubs using two cost structures; the medium and the large truck types. We use the selected intermediate hubs to solve the IP model considering all time-feasible paths with at most one stop, referred to as the 1-stop variant (1S). We solve the path-based and the optimized GIT-based approaches, for 2 and 4-hour discretized GIT, constrained by the set of intermediate hubs but with no restriction on the number of stops of paths, which we refer to as the unrestricted stop path (USP), the unrestricted stop GIT 2H (USG2) and the unrestricted stop GIT 4H (USG4) variants, respectively. We aim to understand how we can further improve the solution cost by allowing more stops on paths in the same network. To compare gap performance results, we solve the optimal LP relaxation of the USP variant to obtain a lower bound of the problem.

In Table 4.4, we report the weighted statistics about the GIT structure and the gap performance of the solutions when using the medium size truck type. Table 4.5 reports path statistics of the solution approaches when using the medium truck type. When we compare the gap performance of the 1S and USP variants of the 6 intermediate hub set against the performance of the 1S and USP variants of the 7 intermediate hub set, we observe a slight improvement in gap performance of about 0.2%, which is expected and desirable given that more intermediate hubs become available. This is also observed for the USG2 and USG4 counterparts which show a very small gap difference of approximately 0.1%. The improvement in gap performance is more noticeable for all the variants of the 8 intermediate hub set, more than 1%, compared to the variants of the 6 and 7 intermediate hub sets. Table 4.5 shows that the percentage of commodity path volume routed over different path lengths is very similar among the USP, USG2 and USG4 variants of the 6, 7 and 8 intermediate hub sets. The IS variants route more commodity volume over paths with 1 stop compared to the other variants because this variant is restricted by the paths composed of at most 1 stop.

In terms of the structure of the solutions, none of them show a 2-hour or higher discretized GIT structure, except for the variants USG2 and USG4, which return a 2-hour and a 4-hour discretized GIT structure by design, respectively. Nevertheless, all solutions across all instances show more than 90% of a 2, 4 and 8-hour discretized GIT structure.

Table 4.4: Weighted Statistics Using the Medium Size Truck Type

	Objective Value	Gap	% GIT 2H	% GIT 4H	% GIT 8H	% GIT 12H	% GIT 16H	% Tree
6 Intermediate Hubs								
1S	1183425	10.37%	95.52%	93.61%	93.18%	89.20%	85.34%	79.14%
USP	1182817	10.31%	93.64%	92.25%	91.39%	88.30%	85.35%	81.47%
USG2	1189221	10.91%	100.00%	96.87%	94.74%	91.93%	88.70%	83.20%
USG4	1189812	10.96%	100.00%	100.00%	97.11%	93.20%	89.25%	83.68%
7 Intermediate Hubs								
1S	1180049	10.05%	95.58%	94.33%	94.14%	89.09%	85.69%	78.50%
USP	1181822	10.22%	95.18%	93.41%	93.31%	89.43%	86.34%	80.86%
USG2	1190201	11.00%	100.00%	97.02%	94.94%	90.60%	87.80%	83.09%
USG4	1190637	11.04%	100.00%	100.00%	98.22%	93.55%	90.63%	85.13%
8 Intermediate Hubs								
1S	1172422	9.34%	96.03%	94.23%	93.17%	87.38%	83.81%	78.82%
USP	1170350	9.15%	93.82%	91.87%	90.67%	86.73%	83.11%	77.30%
USG2	1177933	9.86%	100.00%	97.30%	95.68%	90.77%	86.67%	81.65%
USG4	1177872	9.85%	100.00%	100.00%	96.80%	91.10%	87.64%	82.68%

Table 4.5: Path Statistics Using the Medium Size Truck Type

	Total Nº of paths	% Direct volume	% Volume 1 stop	% Volume 2 stops	% Volume $\geq$ 3 stops	Arc utilization
6 Intermediate Hubs						
1S	26622	62.59%	37.41%	0.00%	0.00%	96.03%
USP	20210	62.99%	35.66%	1.34%	0.01%	95.18%
USG2	20210	62.68%	36.02%	1.28%	0.02%	94.76%
USG4	20210	62.06%	36.62%	1.32%	0.01%	94.76%
7 Intermediate Hubs						
1S	31126	64.36%	35.64%	0.00%	0.00%	95.58%
USP	21066	65.04%	33.79%	1.17%	0.00%	95.23%
USG2	21066	64.07%	34.79%	1.11%	0.02%	94.70%
USG4	21066	64.28%	34.86%	0.86%	0.00%	94.63%
8 Intermediate Hubs						
1S	33077	63.53%	36.47%	0.00%	0.00%	95.91%
USP	22606	64.41%	33.95%	1.61%	0.03%	95.59%
USG2	22606	64.39%	34.35%	1.25%	0.01%	94.96%
USG4	22606	63.89%	34.73%	1.36%	0.02%	94.95%

Similarly, as in the medium truck type experiments, Table 4.6 reports the weighted statistics about the discretized GIT structure and the gap performance of the solutions when using the large size truck type. Table 4.7 reports path statistics of the solution approaches when using the large truck type. In this case, the gap performance of the variants of the 7 intermediate hub set against the performance of the variants of the 8 intermediate hub set is slightly better, more than 1% gap improvement. Also, the gap enhancement of the variants of the 8 intermediate hub set is about 2-3% better than the variants of the 6 intermediate hub set. The 3.42% gap difference between the 1S and USP variants of the 8 intermediate hub set reveals that allowing more stops on paths over the same network has a significant impact on the solution cost for the data set we study.

Table 4.7 shows that, equivalently as we observe in the medium truck type experiments, the percentage of commodity path volume routed over different path lengths is relatively similar among all variants in each intermediate hub set, expect for the 1S case. In terms of the structure of the solutions, none of them return a 2-hour or higher discretized GIT structure, however, they still show more than 90% of a 2, 4 and 8-hour discretized GIT structure.

Table 4.6: Weighted Statistics Using the Large Size Truck Type

	Objective value	Gap	% GIT 2H	% GIT 4H	% GIT 8H	% GIT 12H	% GIT 16H	% Tree
6 Intermediate Hubs								
1S	854833	21.06%	97.72%	96.46%	95.96%	92.74%	88.32%	83.45%
USP	836750	18.50%	95.44%	93.82%	93.51%	91.02%	87.44%	82.33%
USG2	844752	19.63%	100.00%	96.90%	96.30%	93.09%	90.17%	83.67%
USG4	841898	19.23%	100.00%	100.00%	98.00%	95.00%	91.07%	85.34%
7 Intermediate Hubs								
1S	840126	18.98%	95.44%	93.75%	92.82%	88.42%	85.79%	78.58%
USP	826340	17.02%	94.16%	92.25%	91.61%	88.37%	85.97%	81.41%
USG2	835626	18.34%	100.00%	95.07%	93.62%	90.38%	87.61%	83.92%
USG4	836958	18.53%	100.00%	100.00%	96.17%	93.19%	89.41%	84.16%
8 Intermediate Hubs								
1S	839832	18.93%	95.30%	92.99%	91.68%	86.17%	81.99%	73.90%
USP	815660	15.51%	93.30%	91.50%	90.91%	87.40%	83.69%	77.58%
USG2	825026	16.84%	100.00%	95.66%	93.40%	89.35%	85.92%	79.32%
USG4	825369	16.89%	100.00%	100.00%	97.19%	92.54%	89.87%	82.90%

Table 4.7: Path Statistics Using the Large Size Truck Type

	Total Nº of paths	% Direct volume	% Volume 1 stop	% Volume 2 stops	% Volume $\geq$ 3 stops	Arc utilization
6 Intermediate Hubs						
1S	24770	58.30%	41.70%	0.00%	0.00%	91.74%
USP	20393	59.51%	38.10%	2.27%	0.12%	91.21%
USG2	20393	58.75%	38.97%	2.21%	0.07%	90.85%
USG4	20393	58.04%	39.58%	2.31%	0.07%	90.70%
7 Intermediate Hubs						
1S	29175	57.41%	42.59%	0.00%	0.00%	92.93%
USP	22129	58.23%	39.07%	2.59%	0.11%	92.29%
USG2	22129	56.43%	41.02%	2.45%	0.10%	91.61%
USG4	22129	57.02%	40.50%	2.37%	0.11%	91.43%
8 Intermediate Hubs						
1S	33077	58.29%	41.71%	0.00%	0.00%	92.59%
USP	24018	60.12%	37.46%	2.33%	0.09%	92.41%
USG2	24018	58.36%	38.73%	2.89%	0.03%	91.93%
USG4	24018	58.68%	38.99%	2.29%	0.03%	91.57%

Figures 4.5 and 4.6 show the sorted volume in the cities of the 8 intermediate hubs for the 1S and USP variants respectively, when using the large truck type. Both figures illustrate that more than 70% of the sorted volume is concentrated in 4 centered cities. In both variants, the percentage of sorted volume among hubs remains similar. Figure 4.7 depicts the sorted volume taking paths with 2 or more stops for the USP variant, in which we observe that more than half of this volume is sorted in centered cities, however, we see that the percentage of commodity volume taking paths with 2 or more stops passing through more peripheral cities such as 791, 592 and 594 is more significant than the percentage of commodity volume taking paths with one stop.

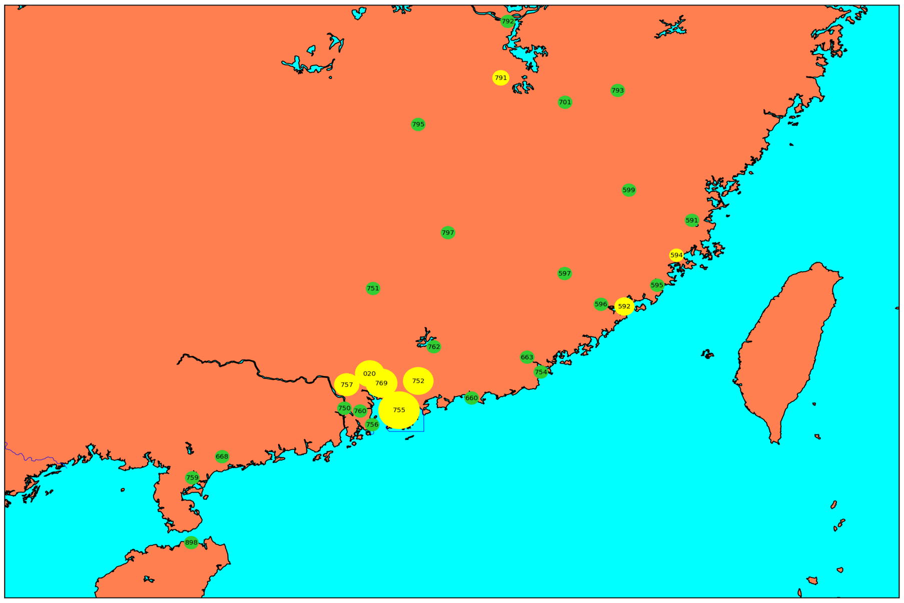

Figure 4.5: Sorted Volume in Cities for the 1S Variant with 8 Intermediate Hubs Using the Large Truck Type

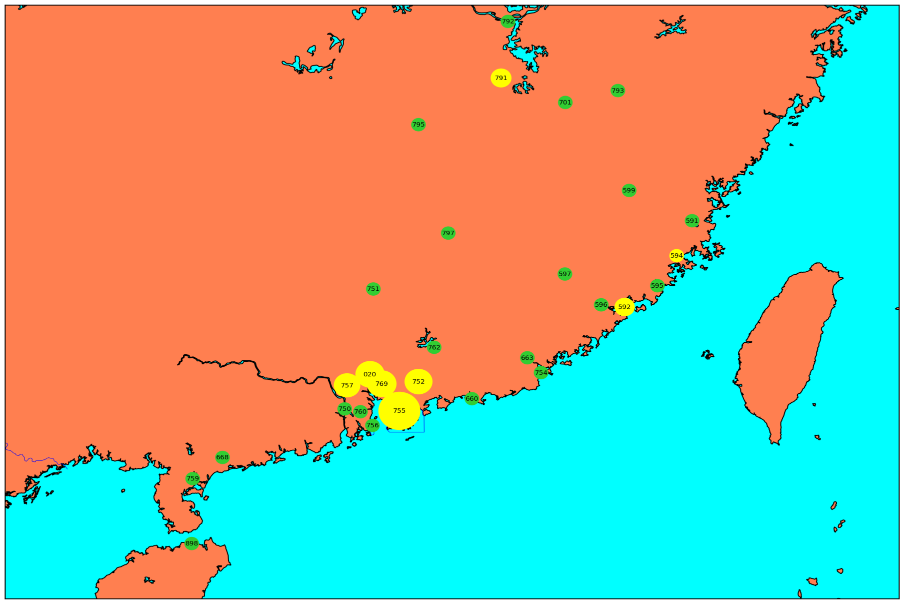

Figure 4.6: Sorted Volume in Cities for the USP Variant with 8 Intermediate Hubs Using the Largest Truck Type

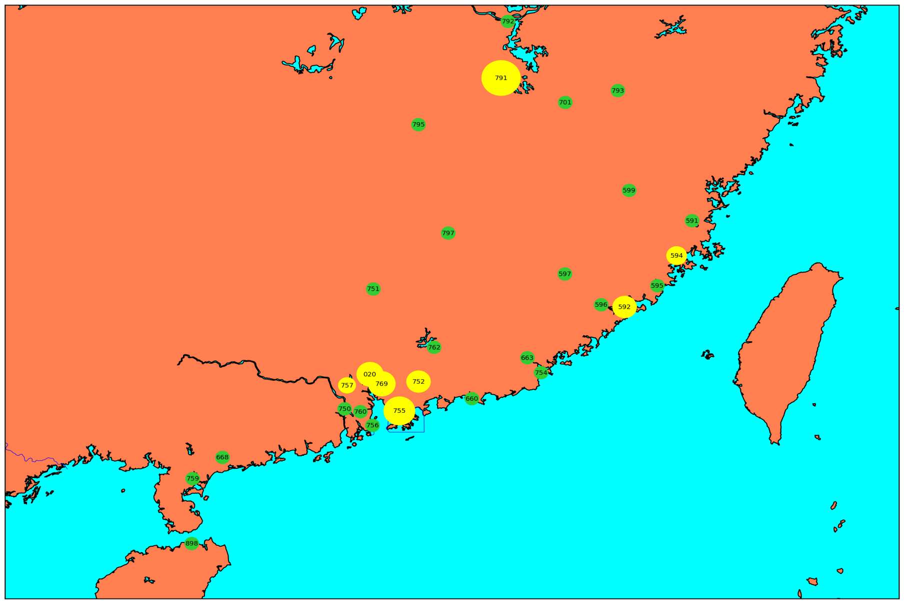

Figure 4.7: Sorted Volume Taking 2 or More Stops in Cities for the USP Variant with 8 Intermediate Hubs Using the Largest Truck Type

4.6.4 Experiments for Small Instances

In this set of experiments, we aim to understand how effective the performance of the approaches are, the structure that the solutions show, and how solutions differ among the approaches. In Table 4.8, we report statistics about the gap performance and discretized GIT structure of the solutions for the small size set of instances. We show whether the solutions show a discretized GIT structure for different discretized GITs: 2, 4, 8, 12, and 16-hour, which we refer to as GIT 2H, GIT 4H, GIT 8H, GIT 12H and GIT 16H respectively, and we also verify whether the solutions show an in-tree structure. Similarly, in Table 4.9, we report the weighted, by commodity volume, discretized GIT statistics. To obtain lower bounds of the problem and measure the gap performance of the solutions of the approaches, we compute the optimal solution of the LP relaxation of the path-based model. Exclusively for this small size set of instances, we solve the path-based model considering all time-feasible paths (only constrained by the time requirement of commodities), which we refer to as the full model, allowing a maximum of 10 hours of running time, in order to obtain tighter lower bounds of the problem. For the larger instances, it is more cumbersome to compute all time-feasible paths and solve a full model directly into an MIP solver. As an example, when having 20 hubs and 500 commodities (close to the size of medium instances), and assuming no restriction on the time requirement to make the computation simple (we need to compute all paths only constrained by time requirements but not by stops), the number of paths with exactly 3 stops is $500 \times 18 \times 17 \times 16$, which is close to 2.5 millions paths.

We observe that the full model, the path-based, the GIT-based and the optimized GIT-based solutions all return a 2-hour discretized GIT structure (the GIT-based and optimized GIT-based approaches are a 2-hour discretized GIT by construction). Furthermore, the path-based, the GIT-based and the optimized GIT-based solutions all show a 4-hour discretized GIT structure across all instances. All geographical instances show an 8, 12, and 16-hour discretized GIT structure, except for the full model solutions of the Geo3 instance (note that a 16-hour discretized GIT is also an 8, 4 and 2-hour discretized GIT, nonetheless, not necessarily a 12-hour discretized GIT). This is expected given the small size of the instances, and therefore, the less complex the network is. Moreover, the path-based, the GIT-based and the optimized GIT-based solutions of the Geo1 and Geo2 instances show an in-tree structure mainly because hubs, and thus the demand associated with them, are geographically dispersed in which the solution is more prone to form an in-tree structure for small networks. In terms of an operational viewpoint, a ”large” hour discretized GIT is simple to operate given that fewer instructions are needed to route packages at a terminal.

We do not report ”pure” GIT structure statistics because all solutions for the small, medium and large size set of instances, show a GIT structure. In other words, in order not to comply with a GIT structure, we need to have at least two commodities with the same remaining available time at a terminal heading to a different next terminal. Since we use rational numbers for travel times on arcs, this rarely occurs.

Since we feed the optimized GIT-based model with the paths generated in the path-based approach and given the small size of these instances, it turns out that almost all instances return the same gap performance for most of the path-based and the optimized GIT-based solutions, except for the Rand2 and Rand4 instances, nevertheless they are still close. The GIT-based solutions are slightly better than the path-based and the optimized GIT-based solutions, except for the Geo4 instance. Half of the instances yield a gap performance of about 2%, and all of the path-based and optimized GIT-based solutions are very close to the gap performance of the full model, around 2% on average. Note that the full model solutions of the Rand4, Geo1, Geo2 and Geo3 instances are optimal solutions given that they end up with 0% of gap.

Table 4.8: Statistics of the Small Size Set of Instances

	Objective value	Gap	% GIT 2H	% GIT 4H	% GIT 8H	% GIT 12H	% GIT 16H	% Tree
Full model								
10-hub Rand1	29675	11.87%	100.0%= $\frac{68}{68}$	100.0%= $\frac{68}{68}$	97.06%= $\frac{66}{68}$	89.71%= $\frac{61}{68}$	86.76%= $\frac{59}{68}$	76.47%= $\frac{52}{68}$
10-hub Rand2	29277	8.07%	100.0%= $\frac{56}{56}$	98.21%= $\frac{55}{56}$	98.21%= $\frac{55}{56}$	98.21%= $\frac{55}{56}$	96.43%= $\frac{54}{56}$	71.43%= $\frac{40}{56}$
10-hub Rand3	39213	2.56%	100.0%= $\frac{47}{47}$	100.0%= $\frac{47}{47}$	97.87%= $\frac{46}{47}$	93.62%= $\frac{44}{47}$	85.11%= $\frac{40}{47}$	59.57%= $\frac{28}{47}$
10-hub Rand4	18155	0.0%	100.0%= $\frac{37}{37}$	100.0%= $\frac{37}{37}$	100.0%= $\frac{37}{37}$	97.3%= $\frac{36}{37}$	97.3%= $\frac{36}{37}$	91.89%= $\frac{34}{37}$
10-hub Geo1	17843	0.0%	100.0%= $\frac{38}{38}$	100.0%= $\frac{38}{38}$	100.0%= $\frac{38}{38}$	100.0%= $\frac{38}{38}$	100.0%= $\frac{38}{38}$	97.37%= $\frac{37}{38}$
10-hub Geo2	20166	0.0%	100.0%= $\frac{45}{45}$	100.0%= $\frac{45}{45}$	100.0%= $\frac{45}{45}$	100.0%= $\frac{45}{45}$	100.0%= $\frac{45}{45}$	95.56%= $\frac{43}{45}$
10-hub Geo3	33889	0.0%	100.0%= $\frac{50}{50}$	100.0%= $\frac{50}{50}$	100.0%= $\frac{50}{50}$	98.0%= $\frac{49}{50}$	98.0%= $\frac{49}{50}$	76.0%= $\frac{38}{50}$
10-hub Geo4	46558	3.43%	100.0%= $\frac{61}{61}$	100.0%= $\frac{61}{61}$	100.0%= $\frac{61}{61}$	100.0%= $\frac{61}{61}$	100.0%= $\frac{61}{61}$	90.16%= $\frac{55}{61}$
Path-based								
10-hub Rand1	29848	12.51%	100.0%= $\frac{64}{64}$	100.0%= $\frac{64}{64}$	100.0%= $\frac{64}{64}$	98.44%= $\frac{63}{64}$	92.19%= $\frac{59}{64}$	79.69%= $\frac{51}{64}$
10-hub Rand2	29347	8.32%	100.0%= $\frac{48}{48}$	100.0%= $\frac{48}{48}$	97.92%= $\frac{47}{48}$	97.92%= $\frac{47}{48}$	93.75%= $\frac{45}{48}$	81.25%= $\frac{39}{48}$
10-hub Rand3	39213	2.56%	100.0%= $\frac{47}{47}$	100.0%= $\frac{47}{47}$	100.0%= $\frac{47}{47}$	100.0%= $\frac{47}{47}$	91.49%= $\frac{43}{47}$	68.09%= $\frac{32}{47}$
10-hub Rand4	18196	0.23%	100.0%= $\frac{34}{34}$	100.0%= $\frac{34}{34}$	100.0%= $\frac{34}{34}$	97.06%= $\frac{33}{34}$	97.06%= $\frac{33}{34}$	97.06%= $\frac{33}{34}$
10-hub Geo1	18932	6.11%	100.0%= $\frac{38}{38}$	100.0%= $\frac{38}{38}$	100.0%= $\frac{38}{38}$	100.0%= $\frac{38}{38}$	100.0%= $\frac{38}{38}$	100.0%= $\frac{38}{38}$
10-hub Geo2	20628	2.29%	100.0%= $\frac{36}{36}$	100.0%= $\frac{36}{36}$	100.0%= $\frac{36}{36}$	100.0%= $\frac{36}{36}$	100.0%= $\frac{36}{36}$	100.0%= $\frac{36}{36}$
10-hub Geo3	34314	1.25%	100.0%= $\frac{52}{52}$	100.0%= $\frac{52}{52}$	100.0%= $\frac{52}{52}$	100.0%= $\frac{52}{52}$	100.0%= $\frac{52}{52}$	88.46%= $\frac{46}{52}$
10-hub Geo4	48692	8.17%	100.0%= $\frac{55}{55}$	100.0%= $\frac{55}{55}$	100.0%= $\frac{55}{55}$	100.0%= $\frac{55}{55}$	100.0%= $\frac{55}{55}$	90.91%= $\frac{50}{55}$
GIT-based								
10-hub Rand1	30573	15.25%	100.0%= $\frac{65}{65}$	100.0%= $\frac{65}{65}$	100.0%= $\frac{65}{65}$	100.0%= $\frac{65}{65}$	100.0%= $\frac{65}{65}$	100.0%= $\frac{65}{65}$
10-hub Rand2	30056	10.94%	100.0%= $\frac{54}{54}$	100.0%= $\frac{54}{54}$	98.15%= $\frac{53}{54}$	98.15%= $\frac{53}{54}$	98.15%= $\frac{53}{54}$	98.15%= $\frac{53}{54}$
10-hub Rand3	40086	4.84%	100.0%= $\frac{47}{47}$	100.0%= $\frac{47}{47}$	100.0%= $\frac{47}{47}$	100.0%= $\frac{47}{47}$	100.0%= $\frac{47}{47}$	100.0%= $\frac{47}{47}$
10-hub Rand4	18244	0.49%	100.0%= $\frac{33}{33}$	100.0%= $\frac{33}{33}$	100.0%= $\frac{33}{33}$	100.0%= $\frac{33}{33}$	96.97%= $\frac{32}{33}$	93.94%= $\frac{31}{33}$
10-hub Geo1	18919	6.03%	100.0%= $\frac{35}{35}$	100.0%= $\frac{35}{35}$	100.0%= $\frac{35}{35}$	100.0%= $\frac{35}{35}$	100.0%= $\frac{35}{35}$	100.0%= $\frac{35}{35}$
10-hub Geo2	21251	5.38%	100.0%= $\frac{36}{36}$	100.0%= $\frac{36}{36}$	100.0%= $\frac{36}{36}$	100.0%= $\frac{36}{36}$	100.0%= $\frac{36}{36}$	100.0%= $\frac{36}{36}$
10-hub Geo3	34617	2.15%	100.0%= $\frac{52}{52}$	100.0%= $\frac{52}{52}$	100.0%= $\frac{52}{52}$	100.0%= $\frac{52}{52}$	100.0%= $\frac{52}{52}$	98.08%= $\frac{51}{52}$
10-hub Geo4	47835	6.27%	100.0%= $\frac{61}{61}$	100.0%= $\frac{61}{61}$	100.0%= $\frac{61}{61}$	100.0%= $\frac{61}{61}$	100.0%= $\frac{61}{61}$	100.0%= $\frac{61}{61}$
Opt GIT-based								
10-hub Rand1	29848	12.51%	100.0%= $\frac{63}{63}$	100.0%= $\frac{63}{63}$	98.41%= $\frac{62}{63}$	96.83%= $\frac{61}{63}$	90.48%= $\frac{57}{63}$	77.78%= $\frac{49}{63}$
10-hub Rand2	29348	8.33%	100.0%= $\frac{52}{52}$	100.0%= $\frac{52}{52}$	98.08%= $\frac{51}{52}$	98.08%= $\frac{51}{52}$	92.31%= $\frac{48}{52}$	76.92%= $\frac{40}{52}$
10-hub Rand3	39213	2.56%	100.0%= $\frac{47}{47}$	100.0%= $\frac{47}{47}$	97.87%= $\frac{46}{47}$	97.87%= $\frac{46}{47}$	91.49%= $\frac{43}{47}$	68.09%= $\frac{32}{47}$
10-hub Rand4	18248	0.51%	100.0%= $\frac{32}{32}$	100.0%= $\frac{32}{32}$	100.0%= $\frac{32}{32}$	100.0%= $\frac{32}{32}$	96.88%= $\frac{31}{32}$	96.88%= $\frac{31}{32}$
10-hub Geo1	18932	6.11%	100.0%= $\frac{38}{38}$	100.0%= $\frac{38}{38}$	100.0%= $\frac{38}{38}$	100.0%= $\frac{38}{38}$	100.0%= $\frac{38}{38}$	100.0%= $\frac{38}{38}$
10-hub Geo2	20628	2.29%	100.0%= $\frac{36}{36}$	100.0%= $\frac{36}{36}$	100.0%= $\frac{36}{36}$	100.0%= $\frac{36}{36}$	100.0%= $\frac{36}{36}$	100.0%= $\frac{36}{36}$
10-hub Geo3	34314	1.25%	100.0%= $\frac{52}{52}$	100.0%= $\frac{52}{52}$	100.0%= $\frac{52}{52}$	100.0%= $\frac{52}{52}$	100.0%= $\frac{52}{52}$	80.77%= $\frac{42}{52}$
10-hub Geo4	48692	8.17%	100.0%= $\frac{57}{57}$	100.0%= $\frac{57}{57}$	100.0%= $\frac{57}{57}$	100.0%= $\frac{57}{57}$	100.0%= $\frac{57}{57}$	91.23%= $\frac{52}{57}$

Table 4.9: Weighted Statistics of the Small Size Set of Instances

	Objective value	Gap	% GIT 2H	% GIT 4H	% GIT 8H	% GIT 12H	% GIT 16H	% Tree
Full model								
10-hub Rand1	29675	11.87%	100.00%	100.00%	82.94%	57.90%	53.81%	47.10%
10-hub Rand2	29277	8.07%	100.00%	96.93%	96.93%	96.93%	95.87%	66.59%
10-hub Rand3	39213	2.56%	100.00%	100.00%	98.45%	85.33%	73.95%	48.42%
10-hub Rand4	18155	0.00%	100.00%	100.00%	100.00%	90.55%	90.55%	87.70%
10-hub Geo1	17843	0.00%	100.00%	100.00%	100.00%	100.00%	100.00%	88.74%
10-hub Geo2	20166	0.00%	100.00%	100.00%	100.00%	100.00%	100.00%	93.41%
10-hub Geo3	33889	0.00%	100.00%	100.00%	100.00%	86.55%	86.55%	53.07%
10-hub Geo4	46558	3.43%	100.00%	100.00%	100.00%	100.00%	100.00%	76.30%
Path-based								
10-hub Rand1	29848	12.51%	100.00%	100.00%	100.00%	96.30%	66.57%	50.50%
10-hub Rand2	29347	8.32%	100.00%	100.00%	95.36%	95.36%	89.75%	78.69%
10-hub Rand3	39213	2.56%	100.00%	100.00%	100.00%	100.00%	89.24%	55.84%
10-hub Rand4	18196	0.23%	100.00%	100.00%	100.00%	62.97%	62.97%	62.97%
10-hub Geo1	18932	6.11%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
10-hub Geo2	20628	2.29%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
10-hub Geo3	34314	1.25%	100.00%	100.00%	100.00%	100.00%	100.00%	78.32%
10-hub Geo4	48692	8.17%	100.00%	100.00%	100.00%	100.00%	100.00%	77.48%
GIT-based								
10-hub Rand1	30573	15.25%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
10-hub Rand2	30056	10.94%	100.00%	100.00%	97.43%	97.43%	97.43%	97.43%
10-hub Rand3	40086	4.84%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
10-hub Rand4	18244	0.49%	100.00%	100.00%	100.00%	100.00%	67.17%	65.40%
10-hub Geo1	18919	6.03%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
10-hub Geo2	21251	5.38%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
10-hub Geo3	34617	2.15%	100.00%	100.00%	100.00%	100.00%	100.00%	99.21%
10-hub Geo4	47835	6.27%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Opt GIT-based								
10-hub Rand1	29848	12.51%	100.00%	100.00%	96.28%	79.91%	60.46%	43.70%
10-hub Rand2	29348	8.33%	100.00%	100.00%	95.10%	95.10%	73.57%	61.29%
10-hub Rand3	39213	2.56%	100.00%	100.00%	98.47%	98.47%	88.96%	53.78%
10-hub Rand4	18248	0.51%	100.00%	100.00%	100.00%	100.00%	64.81%	64.81%
10-hub Geo1	18932	6.11%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
10-hub Geo2	20628	2.29%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
10-hub Geo3	34314	1.25%	100.00%	100.00%	100.00%	100.00%	100.00%	65.89%
10-hub Geo4	48692	8.17%	100.00%	100.00%	100.00%	100.00%	100.00%	79.05%

In Table 4.10, we report statistics about the distribution of the number of stops of commodity path solutions and arc utilization metrics. Table 4.10 shows that all the solutions of the approaches tend to route a similar distribution of the number of stops of commodity paths. Nearly 90% of the commodity volume is routed either over direct paths or paths with 1 stop for all solutions of the approaches. It is expected that when there are not many commodities, more volume is sent over the direct route because there are fewer chances to take advantage of consolidation opportunities. Given that the different approaches return close solution costs for most of the instances, we notice similar commodity volume routed over the same length paths and close arc utilization as well.

Table 4.10: Path Statistics of the Small Size Set of Instances

	Total Nº of paths/GITs	% Direct volume	% Volume 1 stop	% Volume 2 stops	% Volume $\geq$ 3 stops	Arc utilization
Full model						
10-hub Rand1	1825737	81.08%	11.15%	6.22%	1.54%	88.08%
10-hub Rand2	2116984	50.88%	41.23%	7.31%	0.58%	91.33%
10-hub Rand3	737995	84.41%	12.59%	2.51%	0.49%	90.58%
10-hub Rand4	287979	89.79%	5.92%	2.60%	1.69%	76.22%
10-hub Geo1	104482	48.15%	28.81%	18.87%	4.18%	61.12%
10-hub Geo2	349552	40.46%	36.99%	12.10%	10.45%	52.61%
10-hub Geo3	103039	78.16%	15.11%	5.20%	1.54%	84.31%
10-hub Geo4	218371	68.16%	24.14%	5.00%	2.70%	89.20%
Path-based						
10-hub Rand1	790	78.47%	16.15%	5.02%	0.36%	89.21%
10-hub Rand2	721	58.70%	36.01%	4.13%	1.16%	92.85%
10-hub Rand3	812	82.44%	15.86%	0.83%	0.87%	91.24%
10-hub Rand4	295	84.78%	10.43%	2.52%	2.26%	76.20%
10-hub Geo1	267	66.32%	22.46%	5.89%	5.33%	48.16%
10-hub Geo2	282	56.15%	31.59%	6.41%	5.85%	45.05%
10-hub Geo3	459	79.57%	18.47%	1.23%	0.73%	77.56%
10-hub Geo4	497	70.17%	26.24%	2.92%	0.68%	81.87%
GIT-based						
10-hub Rand1	1549	72.45%	19.70%	4.35%	3.50%	83.81%
10-hub Rand2	774	52.25%	26.32%	9.31%	12.12%	90.87%
10-hub Rand3	747	69.91%	22.65%	5.70%	1.75%	88.83%
10-hub Rand4	277	83.75%	8.39%	3.35%	4.50%	77.10%
10-hub Geo1	158	59.99%	27.80%	8.21%	4.00%	49.78%
10-hub Geo2	222	56.27%	30.76%	7.11%	5.85%	45.21%
10-hub Geo3	798	71.09%	20.37%	7.22%	1.32%	78.50%
10-hub Geo4	903	53.99%	30.66%	12.04%	3.32%	86.47%
Opt GIT-based						
10-hub Rand1	790	78.57%	16.31%	4.45%	0.67%	89.05%
10-hub Rand2	721	73.83%	18.25%	5.70%	2.22%	93.06%
10-hub Rand3	812	82.29%	15.82%	1.02%	0.87%	91.50%
10-hub Rand4	295	85.95%	10.22%	1.52%	2.31%	71.91%
10-hub Geo1	267	66.32%	22.46%	5.89%	5.33%	48.16%
10-hub Geo2	282	56.15%	31.59%	6.41%	5.85%	45.05%
10-hub Geo3	459	79.00%	16.89%	3.29%	0.81%	79.32%
10-hub Geo4	497	70.30%	25.12%	3.34%	1.24%	82.77%

Table 4.11 shows all paths generated in the path-based approach. On average, 29% of

the paths generated have 2 or more stops, however, the path-based and the optimized GIT-based solutions only employ 5.73% and 6.37% on average, respectively, of the paths in this category in their final selection. This result is largely explained by the fact that to find high-quality paths of 2 or more stops, many more paths have to be generated compared to 1 stop paths simply because there are exponentially many more combinations of possible stops for longer paths.

Table 4.11: Paths Generated in the Path-based Approach for the Small Size Set of Instances

	% Direct	% 1 stop	% 2 stops	% $\geq$ 3 stops
10-hub Rand1	18.08%	50.94%	26.14%	4.84%
10-hub Rand2	21.71%	39.92%	23.89%	14.49%
10-hub Rand3	22.48%	51.86%	19.49%	6.17%
10-hub Rand4	25.06%	49.93%	22.08%	2.92%
10-hub Geo1	29.60%	46.69%	14.91%	8.80%
10-hub Geo2	20.57%	41.72%	23.61%	14.09%
10-hub Geo3	30.56%	48.45%	19.19%	1.79%
10-hub Geo4	26.30%	42.64%	16.61%	14.46%

4.6.5 Experiments for Medium Instances

In this set of experiments, we aim to understand how effective the approaches are in producing high-quality solutions as we increase the size of the instances, and more importantly, to analyze the impact of the discretized GIT structure on the performance and structure of the solutions compared to the path-based approach. Tables 4.12 and 4.13 report the unweighted and weighted statistics for the medium size set of instances; 20-hub and 30-hub instances. We first observe that none of the path-based solutions show a 2-hour discretized GIT structure, even the optimized GIT-based solutions do not show a 4-hour discretized structure anymore, which is expected for this formulation, whereas the GIT-based solutions still show a 4-hour discretized GIT structure. It is not surprising to see less of discretized GIT structure and an in-tree structure as we increase the size of the instances given that more commodities are passing through or originating at terminals, therefore, more outbound arcs may be activated at terminals that violate the structure of a discretized GIT or an in-tree.

In terms of the performance of the solution of the approaches, instances become harder to solve in comparison to the small size set of instances. The absolute gap shown is computed with respect to the optimal LP relaxation of the path-based approach, therefore, the lower bounds are not as tight as in the small instance set. An example of this is the gap performance of the 20-hub Geo instance. The gap performance difference between the GIT-based solutions and the path-based solutions is more accentuated, resulting in a gap difference of 3.23% on average, and the gap performance difference between the path-based and the optimized GIT-based solutions is relatively small, 0.84% on average. It is intriguing to observe that the optimized GIT-based approach outperforms the GIT-based approach more markedly in these instances. On one hand, the optimized GIT-based approach is optimizing the construction of the discretized GIT structure for each destination altogether in the formulation, and not separately as in the GIT-based approach. On the other hand, the optimized GIT-based approach uses the pool of paths generated in the path-based approach which can be of higher quality compared to the paths of the discretized GITs generated in the GIT-based approach.

Table 4.12: Statistics of the Medium Size Set of Instances

	Objective value	Gap	% GIT 2H	% GIT 4H	% GIT 8H	% GIT 12H	% GIT 16H	% Tree
Path-based								
20-hub Rand	80177	13.91%	97.6%= $\frac{163}{167}$	96.41%= $\frac{161}{167}$	95.21%= $\frac{159}{167}$	94.01%= $\frac{157}{167}$	90.42%= $\frac{151}{167}$	85.63%= $\frac{143}{167}$
20-hub Geo	40073	82.72%	97.97%= $\frac{145}{148}$	97.97%= $\frac{145}{148}$	97.97%= $\frac{145}{148}$	97.3%= $\frac{144}{148}$	95.95%= $\frac{142}{148}$	95.27%= $\frac{141}{148}$
30-hub Rand	292167	6.46%	97.86%= $\frac{457}{467}$	97.0%= $\frac{453}{467}$	95.07%= $\frac{444}{467}$	90.58%= $\frac{423}{467}$	87.15%= $\frac{407}{467}$	76.45%= $\frac{357}{467}$
30-hub Geo	232553	11.51%	97.85%= $\frac{501}{512}$	96.68%= $\frac{495}{512}$	95.51%= $\frac{489}{512}$	90.62%= $\frac{464}{512}$	89.26%= $\frac{457}{512}$	75.39%= $\frac{386}{512}$
GIT-based								
20-hub Rand	82648	16.01%	100.0%= $\frac{169}{169}$	100.0%= $\frac{169}{169}$	100.0%= $\frac{169}{169}$	100.0%= $\frac{169}{169}$	100.0%= $\frac{169}{169}$	100.0%= $\frac{169}{169}$
20-hub Geo	39128	78.39%	100.0%= $\frac{157}{157}$	100.0%= $\frac{157}{157}$	100.0%= $\frac{157}{157}$	100.0%= $\frac{157}{157}$	100.0%= $\frac{157}{157}$	100.0%= $\frac{157}{157}$
30-hub Rand	312070	13.3%	100.0%= $\frac{466}{466}$	100.0%= $\frac{466}{466}$	100.0%= $\frac{466}{466}$	100.0%= $\frac{466}{466}$	100.0%= $\frac{466}{466}$	99.57%= $\frac{464}{466}$
30-hub Geo	250106	19.82%	100.0%= $\frac{542}{542}$	100.0%= $\frac{542}{542}$	99.82%= $\frac{541}{542}$	99.63%= $\frac{540}{542}$	99.63%= $\frac{540}{542}$	99.63%= $\frac{540}{542}$
Opt GIT-based								
20-hub Rand	80469	14.33%	100.0%= $\frac{172}{172}$	98.26%= $\frac{169}{172}$	95.93%= $\frac{165}{172}$	94.19%= $\frac{162}{172}$	91.86%= $\frac{158}{172}$	86.63%= $\frac{149}{172}$
20-hub Geo	40073	82.72%	100.0%= $\frac{150}{150}$	99.33%= $\frac{149}{150}$	99.33%= $\frac{149}{150}$	98.0%= $\frac{147}{150}$	96.0%= $\frac{144}{150}$	94.0%= $\frac{141}{150}$
30-hub Rand	294156	7.18%	100.0%= $\frac{462}{462}$	98.48%= $\frac{455}{462}$	95.45%= $\frac{441}{462}$	92.86%= $\frac{429}{462}$	87.01%= $\frac{402}{462}$	75.11%= $\frac{347}{462}$
30-hub Geo	237178	13.73%	100.0%= $\frac{506}{506}$	98.62%= $\frac{499}{506}$	97.83%= $\frac{495}{506}$	94.47%= $\frac{478}{506}$	91.7%= $\frac{464}{506}$	81.23%= $\frac{411}{506}$

Table 4.13: Weighted Statistics of the Medium Size Set of Instances

	Objective value	Gap	% GIT 2H	% GIT 4H	% GIT 8H	% GIT 12H	% GIT 16H	% Tree
Path-based								
20-hub Rand	80177	13.91%	97.22%	91.21%	90.87%	86.51%	84.70%	82.82%
20-hub Geo	40073	82.72%	94.75%	94.75%	94.75%	93.57%	84.87%	84.76%
30-hub Rand	292167	6.47%	91.04%	89.87%	87.76%	83.60%	79.78%	75.67%
30-hub Geo	232553	11.51%	96.50%	95.37%	92.79%	87.02%	76.88%	62.71%
GIT-based								
20-hub Rand	82648	16.01%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
20-hub Geo	39128	78.39%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
30-hub Rand	312070	13.30%	100.00%	100.00%	100.00%	100.00%	100.00%	99.84%
30-hub Geo	250106	19.82%	100.00%	100.00%	99.54%	99.31%	99.31%	99.31%
Opt GIT-based								
20-hub Rand	80469	14.33%	100.00%	94.58%	88.07%	81.99%	84.66%	77.87%
20-hub Geo	40073	82.72%	100.00%	98.29%	98.29%	95.68%	89.34%	86.31%
30-hub Rand	294156	7.18%	100.00%	99.28%	92.36%	90.26%	81.84%	75.60%
30-hub Geo	237178	13.73%	100.00%	98.51%	97.30%	93.59%	88.99%	81.42%

Table 4.14 shows that the path-based and optimized GIT-based solutions tend to route a similar distribution of the number of stops of commodity paths, in which 92.44% and 92.34% of the commodity volume is routed either over direct paths or paths with one-stop for the path-based and optimized GIT-based solutions, respectively. Here, 2.21% and 1.29% more is sent over paths with 2 or more stops respectively, in comparison to the small size set of instances given that more commodities incentivize taking advantage of more consolidation opportunities. The GIT-based solutions route more than 3% additional commodity volume over paths with 3 or more stops compared to both the path-based and the optimized GIT-based solutions. This tendency of routing a subset of commodities over paths with many stops may help to explain in part the under-performance of the GIT-based approach. Arc utilization remains relatively high, between 87% and 89%, and is similar between the three approaches.

Table 4.14: Path Statistics of the Medium Size Set of Instances

	Total Nº of paths/GITs	% Direct volume	% Volume 1 stop	% Volume 2 stops	% Volume $\geq$ 3 stops	Arc utilization
Path-based						
20-hub Rand	3231	57.48%	36.46%	4.93%	1.14%	95.76%
20-hub Geo	1834	67.85%	17.66%	8.22%	6.28%	70.09%
30-hub Rand	10959	71.10%	24.99%	3.03%	0.89%	96.42%
30-hub Geo	10469	71.74%	22.48%	4.35%	1.44%	94.27%
GIT-based						
20-hub Rand	5376	53.24%	34.44%	8.43%	3.89%	94.74%
20-hub Geo	3888	64.50%	21.03%	8.35%	6.12%	72.36%
30-hub Rand	9355	59.43%	27.19%	8.37%	5.01%	92.94%
30-hub Geo	10499	59.70%	21.95%	11.28%	7.08%	90.87%
Opt GIT-based						
20-hub Rand	3231	57.54%	35.75%	5.69%	1.03%	96.04%
20-hub Geo	1834	67.50%	16.70%	9.99%	5.80%	70.66%
30-hub Rand	10959	68.55%	27.98%	2.77%	0.69%	95.36%
30-hub Geo	10469	72.86%	22.47%	3.68%	0.98%	92.08%

Analogously as we observe in the case of the small size set of instances, Table 4.15 shows that on average, 43% of the paths generated have 2 or more stops, however, both the path-based and the optimized GIT-based solutions only employ 7.56% and 7.66% of paths of these lengths in their final solution, respectively. Given the larger size of the instances, there are many more combinations of possible stops for longer paths compared to the small size set of instances.

Table 4.15: Paths Generated in the Path-based Approach for the Medium Size Set of Instances

	% Direct	% 1 stop	% 2 stops	% $\geq$ 3 stops
20-hub Rand	16.66%	36.95%	29.37%	17.02%
20-hub Geo	15.75%	43.90%	23.68%	16.67%
30-hub Rand	13.51%	44.53%	25.18%	16.78%
30-hub Geo	12.31%	43.30%	26.49%	17.90%

4.6.6 Experiments for Large Instances

In this large size set of experiments, we aim to understand how effectively the approaches behave in more realistic instances. One objective is to measure the penalty cost of imposing a discretized GIT structure in large instances, and also experiment with different ”larger” hour discretized GIT structures on the All-hubs instance to analyze the impact of more rigid discretized GIT structures. In Tables 4.16 and 4.17, we report gap performance and discretized GIT structure statistics for the large size set of instances: 40-hub, 50-hub and All-hubs instances when using medium and large truck types, respectively. Similarly, in Tables 4.18 and 4.19, we report the weighted statistics when using medium and large truck types, respectively.

Path-based solutions do not show a 2-hour discretized GIT structure for medium and large truck types as expected. Tables 4.16 and 4.17 show that the optimized GIT-based solutions show a slightly higher percentage of different discretized GIT structures and an in-tree structure compared to the path-based solutions across all instances for both the medium and the large truck type variants. The structure of the solutions of the GIT-based approach varies enormously compared to the path-based and optimized GIT-based solutions. First, we observe a high percentage of different discretized GIT structures for the GIT-based solutions. This is noticeably seen in the fact that the GIT-based solutions show more than 90% of an in-tree structure compared to the 60%- 70% of an in-tree structure of the path-based and optimized GIT-based solutions. This behaviour is observed for the unweighted and weighted GIT statistics for both the medium and the large truck types.

On the other hand, both the path-based and the optimized-based solutions strongly outperform the GIT-based solutions by 11.58% and 10.24% of gap respectively, for the medium truck type and by 13.86% and 10.97% of gap respectively, for the large truck type. Note that the path-based solutions slightly outperform the optimized GIT-based solutions by 1.3% of gap on average for the medium truck variant and by 2.88% of gap on average for the large truck type variant. It is surprising to observe a striking gap performance difference between the GIT-based and the optimized GIT-based solutions. As we discussed in the medium size set of instances, an explanation of this result can be attributed to the nature of the formulation of the optimized GIT-based approach and to the fact that it uses the paths generated in the path-based approach which may be of higher quality compared to the paths of the discretized GITs generated in the GIT-based approach. Nonetheless, there is another explanation that is related to the GIT-based formulation. When generating GITs in the GIT-based generation heuristic approach, the procedure probably creates high-quality paths for a subset of the commodities and low-quality paths for another subset of paths, thus creating a discretized GIT composed of a mix of these paths in which the model has no option but to select one of them for each destination.

We observe a better gap performance for the medium truck type variant instances compared to the large truck type variant instances because employing a smaller vehicle type favors to more effectively routing commodities, making better use of the vehicle resources, i.e., higher vehicle utilization, and therefore, further reducing solution cost. We also observe this in the 2.99% higher arc utilization on average in the path-based solutions, 3.68% higher arc utilization for the GIT-based solutions and 4.79% higher arc utilization for the optimized GIT-based solutions of the medium truck type variant compared to the large truck type variant (see Tables 4.20 and 4.21). It is noteworthy to mention that the solution cost of the approaches for the large truck type instances are approximately 30% lower on average in comparison to the medium truck type instances. This is largely explained by the fact that the unit cost per capacity of the large vehicle type is 34% lower than the unit cost per capacity of the medium vehicle type.

Table 4.16: Statistics of the Large Size Set of Instances Using the Medium Truck Type

	Objective Value	Gap	% GIT 2H	% GIT 4H	% GIT 8H	% GIT 12H	% GIT 16H	% Tree
Path-based								
40-hub Rand	508600	7.07%	96.81%= $\frac{851}{879}$	95.34%= $\frac{838}{879}$	93.63%= $\frac{823}{879}$	88.05%= $\frac{774}{879}$	84.3%= $\frac{741}{879}$	71.56%= $\frac{629}{879}$
40-hub Geo	374697	8.35%	97.64%= $\frac{787}{806}$	95.41%= $\frac{769}{806}$	93.8%= $\frac{756}{806}$	89.7%= $\frac{723}{806}$	86.85%= $\frac{700}{806}$	77.67%= $\frac{626}{806}$
50-hub Rand	774045	7.25%	97.93%= $\frac{1233}{1259}$	96.74%= $\frac{1218}{1259}$	95.47%= $\frac{1202}{1259}$	91.18%= $\frac{1148}{1259}$	88.01%= $\frac{1108}{1259}$	75.54%= $\frac{951}{1259}$
50-hub Geo	649366	7.25%	97.58%= $\frac{1289}{1321}$	96.21%= $\frac{1271}{1321}$	94.32%= $\frac{1246}{1321}$	90.08%= $\frac{1190}{1321}$	86.68%= $\frac{1145}{1321}$	75.02%= $\frac{991}{1321}$
All-hubs	1159139	8.1%	97.06%= $\frac{2012}{2073}$	95.66%= $\frac{1983}{2073}$	93.87%= $\frac{1946}{2073}$	90.21%= $\frac{1870}{2073}$	86.3%= $\frac{1789}{2073}$	74.77%= $\frac{1550}{2073}$
GIT-based								
40-hub Rand	555612	16.39%	100.0%= $\frac{908}{908}$	99.78%= $\frac{906}{908}$	99.12%= $\frac{900}{908}$	99.12%= $\frac{900}{908}$	99.12%= $\frac{900}{908}$	98.9%= $\frac{898}{908}$
40-hub Geo	414335	19.64%	100.0%= $\frac{858}{858}$	100.0%= $\frac{858}{858}$	99.88%= $\frac{857}{858}$	99.77%= $\frac{856}{858}$	99.42%= $\frac{853}{858}$	98.48%= $\frac{845}{858}$
50-hub Rand	855658	18.05%	100.0%= $\frac{1344}{1344}$	99.78%= $\frac{1341}{1344}$	99.11%= $\frac{1332}{1344}$	98.96%= $\frac{1330}{1344}$	98.96%= $\frac{1330}{1344}$	98.14%= $\frac{1319}{1344}$
50-hub Geo	734338	20.88%	100.0%= $\frac{1409}{1409}$	99.65%= $\frac{1404}{1409}$	99.5%= $\frac{1402}{1409}$	99.5%= $\frac{1402}{1409}$	99.36%= $\frac{1400}{1409}$	98.79%= $\frac{1392}{1409}$
All-hubs	1303157	20.94%	100.0%= $\frac{2169}{2169}$	99.31%= $\frac{2154}{2169}$	98.66%= $\frac{2140}{2169}$	98.43%= $\frac{2135}{2169}$	98.29%= $\frac{2132}{2169}$	97.69%= $\frac{2119}{2169}$
Opt GIT-based								
40-hub Rand	514659	8.35%	100.0%= $\frac{878}{878}$	98.63%= $\frac{866}{878}$	96.24%= $\frac{845}{878}$	90.09%= $\frac{791}{878}$	88.04%= $\frac{773}{878}$	75.85%= $\frac{666}{878}$
40-hub Geo	383022	10.76%	100.0%= $\frac{812}{812}$	97.54%= $\frac{792}{812}$	95.94%= $\frac{779}{812}$	92.98%= $\frac{755}{812}$	90.02%= $\frac{731}{812}$	80.67%= $\frac{655}{812}$
50-hub Rand	780447	8.13%	100.0%= $\frac{1283}{1283}$	97.82%= $\frac{1255}{1283}$	96.1%= $\frac{1233}{1283}$	91.11%= $\frac{1169}{1283}$	87.61%= $\frac{1124}{1283}$	76.38%= $\frac{980}{1283}$
50-hub Geo	665029	10.76%	100.0%= $\frac{1325}{1325}$	98.42%= $\frac{1304}{1325}$	96.3%= $\frac{1276}{1325}$	93.13%= $\frac{1234}{1325}$	87.55%= $\frac{1160}{1325}$	76.91%= $\frac{1019}{1325}$
All-hubs	1154249	7.65%	100.0%= $\frac{2054}{2054}$	98.0%= $\frac{2013}{2054}$	96.35%= $\frac{1979}{2054}$	92.6%= $\frac{1902}{2054}$	90.07%= $\frac{1850}{2054}$	79.5%= $\frac{1633}{2054}$

Table 4.17: Statistics of the Large Size Set of Instances Using the Large Truck Type

	Objective value	Gap	% GIT 2H	% GIT 4H	% GIT 8H	% GIT 12H	% GIT 16H	% Tree
Path-based								
40-hub Rand	349708	11.80%	95.51%= $\frac{851}{891}$	93.83%= $\frac{836}{891}$	92.82%= $\frac{827}{891}$	86.98%= $\frac{775}{891}$	84.62%= $\frac{754}{891}$	73.51%= $\frac{655}{891}$
40-hub Geo	260779	14.51%	95.73%= $\frac{784}{819}$	92.55%= $\frac{758}{819}$	91.09%= $\frac{746}{819}$	86.2%= $\frac{706}{819}$	83.76%= $\frac{686}{819}$	73.26%= $\frac{600}{819}$
50-hub Rand	535869	12.74%	95.35%= $\frac{1211}{1270}$	93.62%= $\frac{1189}{1270}$	91.5%= $\frac{1162}{1270}$	87.95%= $\frac{1117}{1270}$	84.41%= $\frac{1072}{1270}$	72.2%= $\frac{917}{1270}$
50-hub Geo	446851	12.07%	96.53%= $\frac{1336}{1384}$	93.5%= $\frac{1294}{1384}$	91.33%= $\frac{1264}{1384}$	87.36%= $\frac{1209}{1384}$	84.32%= $\frac{1167}{1384}$	74.06%= $\frac{1025}{1384}$
All-hubs	796805	12.84%	97.03%= $\frac{2029}{2091}$	95.22%= $\frac{1991}{2091}$	93.97%= $\frac{1965}{2091}$	89.72%= $\frac{1876}{2091}$	87.14%= $\frac{1822}{2091}$	78.19%= $\frac{1635}{2091}$
GIT-based								
40-hub Rand	393341	25.12%	100.0%= $\frac{911}{911}$	99.56%= $\frac{907}{911}$	99.01%= $\frac{902}{911}$	98.9%= $\frac{901}{911}$	98.9%= $\frac{901}{911}$	98.46%= $\frac{897}{911}$
40-hub Geo	295996	29.79%	100.0%= $\frac{870}{870}$	100.0%= $\frac{870}{870}$	100.0%= $\frac{870}{870}$	99.66%= $\frac{867}{870}$	99.54%= $\frac{866}{870}$	98.97%= $\frac{861}{870}$
50-hub Rand	610487	27.89%	100.0%= $\frac{1343}{1343}$	99.48%= $\frac{1336}{1343}$	98.81%= $\frac{1327}{1343}$	98.66%= $\frac{1325}{1343}$	98.51%= $\frac{1323}{1343}$	96.87%= $\frac{1301}{1343}$
50-hub Geo	527067	31.75%	100.0%= $\frac{1421}{1421}$	99.72%= $\frac{1417}{1421}$	99.44%= $\frac{1413}{1421}$	99.37%= $\frac{1412}{1421}$	99.01%= $\frac{1407}{1421}$	98.31%= $\frac{1397}{1421}$
All-hubs	931411	31.25%	100.0%= $\frac{2201}{2201}$	99.14%= $\frac{2182}{2201}$	98.23%= $\frac{2162}{2201}$	98.09%= $\frac{2159}{2201}$	97.82%= $\frac{2153}{2201}$	96.96%= $\frac{2134}{2201}$
Opt GIT-based								
40-hub Rand	357813	14.39%	100.0%= $\frac{888}{888}$	98.09%= $\frac{871}{888}$	95.83%= $\frac{851}{888}$	91.44%= $\frac{812}{888}$	87.84%= $\frac{780}{888}$	77.59%= $\frac{689}{888}$
40-hub Geo	266210	16.89%	100.0%= $\frac{816}{816}$	98.16%= $\frac{801}{816}$	95.83%= $\frac{782}{816}$	92.89%= $\frac{758}{816}$	90.69%= $\frac{740}{816}$	82.6%= $\frac{674}{816}$
50-hub Rand	549263	15.56%	100.0%= $\frac{1279}{1279}$	98.59%= $\frac{1261}{1279}$	97.03%= $\frac{1241}{1279}$	93.04%= $\frac{1190}{1279}$	89.44%= $\frac{1144}{1279}$	78.11%= $\frac{999}{1279}$
50-hub Geo	464828	16.57%	100.0%= $\frac{1361}{1361}$	98.24%= $\frac{1337}{1361}$	96.62%= $\frac{1315}{1361}$	92.8%= $\frac{1263}{1361}$	89.05%= $\frac{1212}{1361}$	79.5%= $\frac{1082}{1361}$
All-hubs	811983	14.99%	100.0%= $\frac{2082}{2082}$	97.65%= $\frac{2033}{2082}$	95.97%= $\frac{1998}{2082}$	92.27%= $\frac{1921}{2082}$	88.18%= $\frac{1836}{2082}$	79.06%= $\frac{1646}{2082}$

Table 4.18: Weighted Statistics of the Large Size Set of Instances Using the Medium Truck Type

	Objective value	Gap	% GIT 2H	% GIT 4H	% GIT 8H	% GIT 12H	% GIT 16H	% Tree
Path-based								
40-hub Rand	508600	7.07%	87.18%	85.65%	82.09%	77.17%	71.74%	62.36%
40-hub Geo	374697	8.35%	93.45%	89.04%	86.36%	81.31%	76.13%	69.19%
50-hub Rand	774045	7.25%	93.67%	90.39%	88.82%	84.85%	80.19%	68.44%
50-hub Geo	649366	7.25%	95.14%	90.93%	89.08%	82.45%	77.55%	70.02%
All-hubs	1159139	8.1%	93.07%	91.55%	89.39%	86.03%	81.57%	73.10%
GIT-based								
40-hub Rand	555612	16.39%	100.00%	99.31%	98.54%	98.54%	98.54%	98.50%
40-hub Geo	414335	19.64%	100.00%	100.00%	99.28%	99.25%	98.65%	98.29%
50-hub Rand	855658	18.05%	100.00%	98.80%	97.86%	97.65%	97.65%	97.47%
50-hub Geo	734338	20.88%	100.00%	95.95%	95.17%	95.17%	95.07%	94.88%
All-hubs	1303157	20.94%	100.00%	96.50%	95.11%	94.89%	94.84%	94.50%
Opt GIT-based								
40-hub Rand	514659	8.35%	100.00%	94.18%	90.25%	83.32%	80.35%	72.23%
40-hub Geo	383022	10.76%	100.00%	96.09%	93.99%	89.22%	87.61%	78.23%
50-hub Rand	780447	8.13%	100.00%	96.20%	93.86%	89.29%	84.32%	76.35%
50-hub Geo	665029	9.83%	100.00%	94.82%	92.39%	88.68%	81.69%	71.28%
All-hubs	1154249	7.65%	100.00%	95.11%	92.66%	88.71%	85.37%	77.07%

Table 4.19: Weighted Statistics of the Large Size Set of Instances Using the Large Truck Type

	Objective value	Gap	% GIT 2H	% GIT 4H	% GIT 8H	% GIT 12H	% GIT 16H	% Tree
Path-based								
40-hub Rand	349708	11.80%	88.34%	84.58%	82.12%	73.71%	70.47%	63.17%
40-hub Geo	260779	14.51%	91.09%	86.49%	83.68%	78.32%	75.77%	68.46%
50-hub Rand	535869	12.74%	88.08%	85.29%	82.77%	79.78%	76.25%	68.30%
50-hub Geo	446851	12.07%	89.44%	81.46%	76.59%	71.91%	69.52%	60.37%
All-hubs	796805	12.84%	92.51%	88.42%	87.15%	80.75%	78.03%	70.14%
GIT-based								
40-hub Rand	393341	25.12%	100.00%	98.41%	96.70%	96.48%	96.48%	96.33%
40-hub Geo	295996	29.79%	100.00%	100.00%	100.00%	98.26%	97.38%	97.19%
50-hub Rand	610487	27.89%	100.00%	97.26%	95.94%	95.86%	95.66%	95.22%
50-hub Geo	527067	31.75%	100.00%	96.70%	95.26%	94.91%	94.15%	93.68%
All-hubs	931411	31.25%	100.00%	94.11%	91.08%	90.94%	90.74%	90.47%
Opt GIT-based								
40-hub Rand	357813	14.39%	100.00%	93.83%	87.43%	84.21%	78.86%	72.03%
40-hub Geo	266210	16.89%	100.00%	97.24%	92.87%	90.15%	88.47%	82.19%
50-hub Rand	549263	15.56%	100.00%	97.00%	94.72%	90.32%	86.15%	76.12%
50-hub Geo	464828	16.57%	100.00%	97.13%	95.44%	89.66%	84.81%	77.68%
All-hubs	811983	14.99%	100.00%	94.94%	92.03%	87.93%	83.46%	77.33%

In Table 4.20, we observe that the optimized GIT-based solutions tend to route a similar percentage of commodity path volume compared to the path-based solutions for both the medium and the large truck type variants. Nonetheless, equivalently as we observe in the medium size set of instances, the GIT-based solutions route, on average, more than 11% more commodity volume through paths with 2 stops, and more than 9% for paths with 3 or more stops compared to both the path-based and the optimized GIT-based solutions for the medium truck type (and more than 9% and 15% respectively, for the large truck type). Arc utilization remains relatively high and similar between the three approaches, close to 90% for the large truck type and more than 90% for the medium truck type.

The nearness in the gap performance of the optimized GIT-based and path-based solutions is largely explained by two facts; first, a 2-hour discretized GIT structure is a non-highly constrained setting, compared to the relaxation of the path-based formulation, which enables us to achieve very close gap performance solutions among the approaches. And second, the optimized GIT-based approach is fed by the paths generated in the path-based approach, thus, we expect to have a similar distribution in the number of stops of the commodity paths when imposing a 2-hour discretized GIT structure. To see the latter, consider the extreme case that we build a fine-granular discretized GIT structure which allows that almost any path-based solution forms a discretized GIT structure. We observe in Tables 4.20 and 4.21 that more than 94%, on average, of the commodity volume is sent either over the direct route or paths with one stop for both the path-based and the optimized GIT-based solutions across all instances. Commodity paths of the path-based and optimized GIT-based solutions having 2 stops range between 2% and 4% of the commodity volume, and commodity paths with 3 or more stops range between 1% and 2% of the commodity volume. This is reasonable because most of the consolidation happens in a few geographically centered hubs, and thereby taking longer paths that have stops at other intermediate hubs would only increase operational costs.

Table 4.20: Path Statistics of the Large Size Set of Instances Using the Medium Truck Type

	Total Nº of paths/GITs	% Direct volume	% Volume 1 stop	% Volume 2 stops	% Volume $\geq$ 3 stops	Arc utilization
Path-based						
40-hub Rand	22280	63.45%	32.15%	3.87%	0.53%	96.59%
40-hub Geo	18882	71.12%	23.70%	4.22%	0.96%	94.98%
50-hub Rand	35389	68.39%	28.59%	2.56%	0.46%	95.35%
50-hub Geo	37722	68.09%	27.06%	4.13%	0.73%	95.81%
All-hubs	63235	64.51%	30.78%	4.10%	0.60%	95.69%
GIT-based						
40-hub Rand	13012	48.03%	27.52%	15.88%	8.56%	93.99%
40-hub Geo	13283	57.04%	21.60%	8.45%	12.91%	92.16%
50-hub Rand	15691	43.68%	24.23%	17.45%	14.64%	93.97%
50-hub Geo	16934	43.57%	27.88%	18.06%	10.48%	93.56%
All-hubs	21410	38.07%	31.05%	15.25%	15.63%	94.31%
Opt GIT-based						
40-hub Rand	22280	65.45%	30.18%	3.94%	0.42%	95.38%
40-hub Geo	18882	72.54%	21.83%	4.76%	0.87%	92.59%
50-hub Rand	35389	69.14%	28.10%	2.25%	0.50%	94.39%
50-hub Geo	37722	69.14%	26.32%	3.76%	0.79%	94.06%
All-hubs	63235	65.05%	30.94%	3.50%	0.51%	94.81%

Table 4.21: Path Statistics of the Large Size Set of Instances Using the Large Truck Type

	Total Nº of paths/GITs	% Direct volume	% Volume 1 stop	% Volume 2 stops	% Volume $\geq$ 3 stops	Arc utilization
Path-based						
40-hub Rand	25602	61.66%	31.37%	6.01%	0.96%	94.41%
40-hub Geo	24909	65.00%	24.97%	6.56%	3.47%	91.99%
50-hub Rand	43433	63.52%	30.74%	4.70%	1.04%	92.49%
50-hub Geo	44938	64.32%	27.70%	6.63%	1.34%	92.83%
All-hubs	75804	60.81%	32.75%	5.26%	1.18%	92.24%
GIT-based						
40-hub Rand	12912	43.52%	28.36%	15.14%	12.97%	90.35%
40-hub Geo	13250	48.50%	24.79%	12.72%	13.99%	87.08%
50-hub Rand	15765	38.66%	28.19%	14.57%	18.58%	90.73%
50-hub Geo	16846	43.91%	24.74%	16.11%	15.25%	89.88%
All-hubs	21282	29.47%	29.14%	18.29%	23.09%	91.55%
Opt GIT-based						
40-hub Rand	25602	61.59%	30.95%	6.26%	1.20%	91.85%
40-hub Geo	24909	68.13%	23.53%	6.06%	2.28%	88.13%
50-hub Rand	43433	64.31%	30.60%	4.04%	1.05%	89.82%
50-hub Geo	44938	65.59%	27.94%	5.05%	1.42%	89.64%
All-hubs	75804	61.14%	32.33%	5.27%	1.26%	88.94%

Tables 4.22 and 4.23 show that on average, 50% and 55% of the paths generated have 2 or more stops, for the medium truck type and the large truck type variants, respectively. Nevertheless, the solutions of the path-based and the optimized GIT-based model employ on average about 4% and 7% of these paths in their final solutions for the medium and the large truck types, respectively. Equivalently as we see in the medium size set of instances, in order to find high-quality paths of 2 or more stops, many more paths have to be generated compared to one-stop paths, and more prominently in this large set of instances in which an enormous number of more options for stops of the paths are feasible.

Table 4.22: Paths Generated in the Path-based Approach when Using the Medium Truck Type

	% Direct	% 1 stop	% 2 stops	% $\geq$ 3 stops
40-hub Rand	12.62%	39.48%	27.75%	20.14%
40-hub Geo	12.46%	40.04%	27.31%	20.19%
50-hub Rand	11.52%	39.27%	27.60%	21.61%
50-hub Geo	10.99%	39.18%	27.77%	22.06%
All-hubs	10.07%	37.99%	26.33%	25.61%

Table 4.23: Paths Generated in the Path-based Approach when Using the Large Truck Type

	% Direct	% 1 stop	% 2 stops	% $\geq$ 3 stops
40-hub Rand	11.43%	35.98%	28.45%	24.14%
40-hub Geo	9.89%	35.81%	28.20%	26.10%
50-hub Rand	9.43%	35.19%	29.29%	26.09%
50-hub Geo	9.57%	34.71%	29.34%	26.39%
All-hubs Rand	8.64%	35.34%	28.01%	28.00%

In order to measure the impact, on the structure and performance of solutions, of using a ”larger” hour discretized GIT, we experiment, in addition to the 2-hour discretized GIT, with 4, 8, 12, 16 and 72-hour discretized GIT variants of the optimized GIT-based approach for the All-hubs instance using the large truck type. Tables 4.24 and 4.25 report the unweighted and weighted GIT statistics, respectively. Both tables show that a 4-hour discretized GIT returns a very close solution, in terms of gap performance, compared to the 2-hour discretized GIT solutions. In fact, the gap performance of the 4-hour discretized GIT is slightly better than the 2-hour discretized GIT, which may be attributed to the fact that models do not finish optimally (we allow a maximum timelimit of one hour of computation), and moreover the 4-hour discretized GIT is a slightly less difficult problem to solve than a 2-hour discretized GIT. We observe that the gap performance solutions of the 8 and 12-hour discretized GIT, are approximately 1% and 2% more costly than the 2-hour discretized GIT solutions, respectively. Interestingly, the gap performance of the solutions of the 16 and 72-hour discretized GIT are very close, less than 0.1%, to the solutions of the 12-hour discretized GIT. Since the maximum time requirement of commodities is 72 hours, then the 72-hour discretized GIT returns an in-tree structure for each destination. It is intriguing to observe that the difference in gap performance between the path-based and the 72-hour discretized GIT solutions is 3.44%.

Table 4.24: Statistics for Different Discretized GIT of the Optimized GIT-based Approach for the All-hubs Instance when Using the Large Truck Type

	Objective value	Gap	% GIT 2H	% GIT 4H	% GIT 8H	% GIT 12H	% GIT 16H	% Tree
Path-based	796805	12.84%	97.03%= $\frac{2029}{2091}$	95.22%= $\frac{1991}{2091}$	93.97%= $\frac{1965}{2091}$	89.72%= $\frac{1876}{2091}$	87.14%= $\frac{1822}{2091}$	78.19%= $\frac{1635}{2091}$
GIT 2H	811983	14.99%	100.0%= $\frac{2082}{2082}$	97.65%= $\frac{2033}{2082}$	95.97%= $\frac{1998}{2082}$	92.27%= $\frac{1921}{2082}$	88.18%= $\frac{1836}{2082}$	79.06%= $\frac{1646}{2082}$
GIT 4H	809501	14.64%	100.0%= $\frac{2111}{2111}$	100.0%= $\frac{2111}{2111}$	96.83%= $\frac{2044}{2111}$	91.66%= $\frac{1935}{2111}$	88.68%= $\frac{1872}{2111}$	76.84%= $\frac{1622}{2111}$
GIT 8H	817078	15.71%	100.0%= $\frac{2108}{2108}$	100.0%= $\frac{2108}{2108}$	100.0%= $\frac{2108}{2108}$	94.78%= $\frac{1998}{2108}$	90.32%= $\frac{1904}{2108}$	78.46%= $\frac{1654}{2108}$
GIT 12H	820339	16.17%	100.0%= $\frac{2086}{2086}$	100.0%= $\frac{2086}{2086}$	99.66%= $\frac{2079}{2086}$	100.0%= $\frac{2086}{2086}$	91.71%= $\frac{1913}{2086}$	80.97%= $\frac{1689}{2086}$
GIT 16H	820430	16.19%	100.0%= $\frac{2120}{2120}$	100.0%= $\frac{2120}{2120}$	100.0%= $\frac{2120}{2120}$	97.41%= $\frac{2065}{2120}$	100.0%= $\frac{2120}{2120}$	85.75%= $\frac{1818}{2120}$
GIT 72H	821109	16.28%	100.0%= $\frac{2093}{2093}$	100.0%= $\frac{2093}{2093}$	100.0%= $\frac{2093}{2093}$	100.0%= $\frac{2093}{2093}$	100.0%= $\frac{2093}{2093}$	100.0%= $\frac{2093}{2093}$

Table 4.25: Weighted Statistics for Different Discretized GIT of the Optimized GIT-based Approach for the All-hubs Instance when Using the Large Truck Type

	Objective value	Gap	% GIT 2H	% GIT 4H	% GIT 8H	% GIT 12H	% GIT 16H	% Tree
Path-based	796805	12.84%	92.51%	88.42%	87.15%	80.75%	78.03%	70.14%
GIT 2H	811983	14.99%	100.00%	94.94%	92.03%	87.93%	83.46%	77.33%
GIT 4H	809501	14.64%	100.00%	100.00%	93.69%	86.51%	84.34%	74.73%
GIT 8H	817078	15.71%	100.00%	100.00%	100.00%	93.41%	88.42%	79.16%
GIT 12H	820339	16.17%	100.00%	100.00%	99.37%	100.00%	89.34%	81.90%
GIT 16H	820430	16.19%	100.00%	100.00%	100.00%	98.46%	100.00%	86.21%
GIT 72H	821109	16.28%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

Table 4.26 shows that the solutions of different discretized GIT runs are relatively similar in the percentage of commodity path volume routed over different path lengths, in general no more than 2% of difference for each path length. The path-based solutions behave in a similar manner, nonetheless, showing a 2-3% higher arc utilization compared to the counterpart discretized GIT solutions, which may explain in part the better gap performance of the path-based solutions.

Table 4.26: Path Statistics for Different Discretized GIT of the Optimized GIT-based Approach for the All-hubs Instance when Using the Large Truck Type

	Total Nº of paths	% Direct volume	% Volume 1 stop	% Volume 2 stops	% Volume $\geq$ 3 stops	Arc utilization
Path-based	75804	60.81%	32.75%	5.26%	1.18%	92.24%
GIT 2H	75804	61.14%	32.33%	5.27%	1.26%	88.94%
GIT 4H	75804	60.13%	33.24%	5.33%	1.30%	89.97%
GIT 8H	75804	60.52%	32.49%	5.75%	1.24%	89.28%
GIT 12H	75804	60.76%	32.63%	5.36%	1.25%	89.65%
GIT 16H	75804	60.52%	32.66%	5.82%	1.00%	88.93%
GIT 72H	75804	59.65%	32.55%	6.60%	1.21%	89.73%

In Table 4.27 we show computation time statistics for the GIT-based and the path-based approaches that involve the generation of paths and GITs. For each metric, we compute the average over 10 runs. We observe that for the path-based approach, most of the time, 89%, is spent in solving IP models, whereas for the GIT-based approach, most of the time, 61%, is devoted generating GITs. This is attributed to the fact that in the GIT-based approach we generate as many as 3 GITs per destination on each iteration, in which we sample more lower bounds on truck variables in order to improve solution costs faster. Given the fewer number of destination, then there are fewer number of variables we add per destination, in comparison with the 5297 number of commodities in the path-based approach (and consequently more number of variables able to add on each iteration). To observe this, the total number of paths generated is almost 4 times the number of GITs generated even when we generate more GITs per destination on each iteration.

Table 4.27: Average Computational Statistics for The All-hubs Instance

	Paths/GITs	Time (hr)	Solving IPs (Hr)	Generating paths/GITs (Hr)	% Solving IPs	% Generating paths/GITs	Gap IPs
Path-based	78574	8.33	7.44	0.89	89%	11%	13.27%
GIT-based	21327	17.65	6.84	10.81	39%	61%	15.97%

In Tables 4.28 and 4.29, we report the 10 most used cities as an intermediate stop for the path-based and optimized GIT-based solutions, respectively, when using the largest truck type. Figures 4.8 and 4.9 depict the cities of the sorted volume associated with the 10 most used intermediate hubs. Red color cities indicate that more than 5% of commodity volume is sorted in those cities. We observe that in both approach solutions, the same cities, 755 and 020, concentrate approximately 50% of the sorted volume. It is expected to observe the same intermediate hubs having similar sorted volume given that the difference in gap performance of the solutions of the approaches is about 2%, and furthermore, the optimized GIT-based approach employs the same set of paths generated in the path-based approach. We also see that more than 70% of the sorted volume is concentrated in centered cities, for both approaches, due to the fact that most of the commodity volume comes from and heads to those centered cities.

Table 4.28: 10 Most Used Intermediate Cities for the Path-based Solution of the All-hubs Instance Using the Large Truck Type

City	% Sorted Volume
755	31.08%
020	18.30%
769	8.36%
663	7.66%
757	7.01%
791	3.84%
752	3.30%
592	3.19%
760	2.42%
595	2.21%

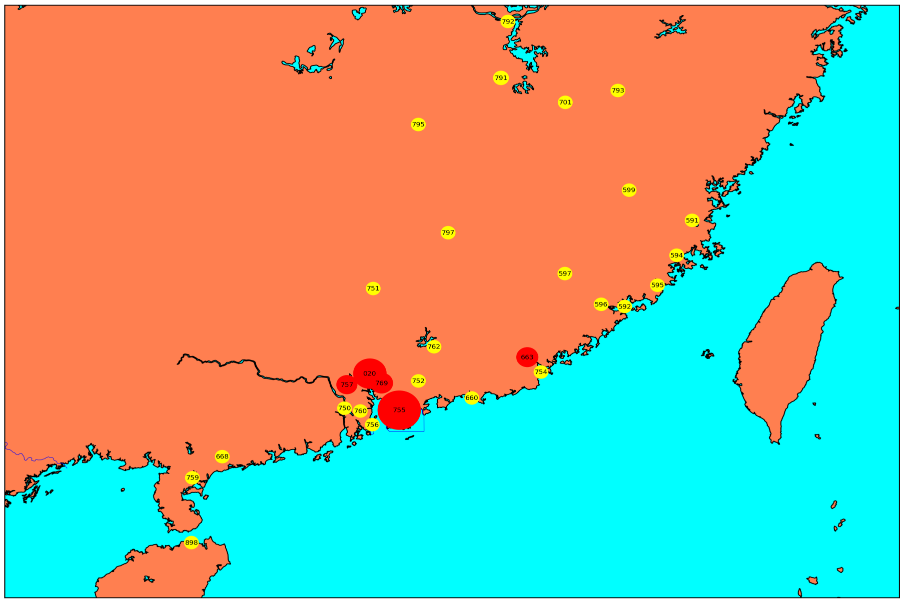

Figure 4.8: Sorted Volume in Cities for the Path-based Solution of the All-hubs Instance Using the Large Truck Type

Table 4.29: 10 Most Used Intermediate Cities for the optimized GIT-based Solution of the All-hubs Instance Using the Large Truck Type

City	% Sorted Volume
755	29.73%
020	19.91%
663	9.64%
769	8.53%
757	6.09%
791	4.64%
592	2.85%
752	2.75%
595	1.91%
762	1.68%

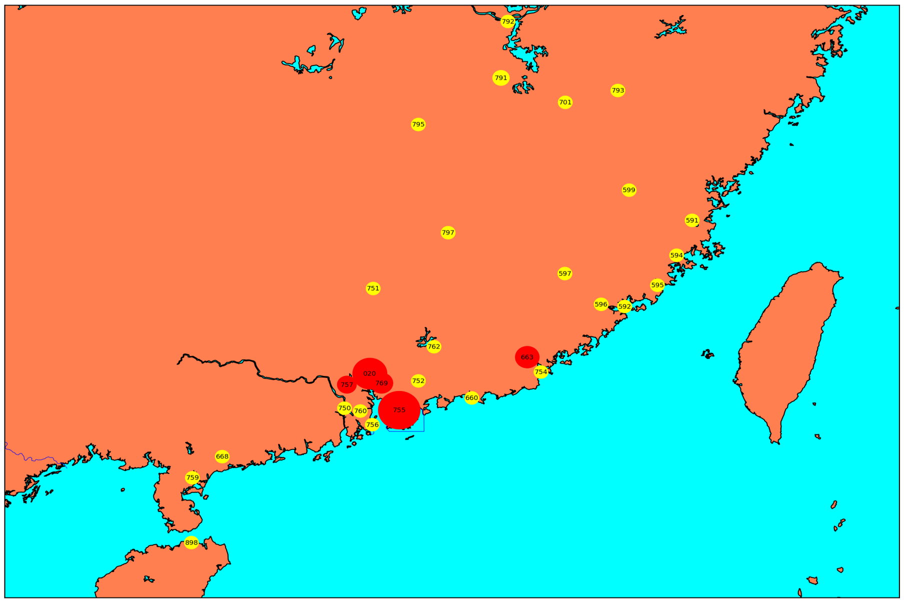

Figure 4.9: Sorted Volume in Cities for the optimized GIT-based Solution of the All-hubs Instance Using the Large Truck Type

4.7 Conclusions and Future Research

In this work, we introduce the concept of GIT and discretized GIT structures, which provides useful operational benefits in practice. We develop a GIT-based, an optimized GIT-based approach and a relaxation of the problem through a path-based formulation.

We propose a dynamic path generation methodology to create multi-stop paths and avoid enumerating all time-feasible paths, which is prohibited to generate for medium and large problems.

We develop three approaches: a relaxation path-based approach, a GIT-based approach, and an optimized GIT-based formulation. The exploration of multi-stop path solutions, compared to one-stop paths, restricted to the set of intermediate hubs selected with the hub selection model of Chapter 3, permits us to improve solution costs by approximately 1% and 3% when using the medium and the large size truck types, respectively.

We demonstrate, via a computational study, that the path-based solutions show a 2-hour and a 4-hour discretized GIT structure, and the gap performance among the three approaches is comparatively similar for the small size set of instances.

For the medium size set of instances, the path-based solutions do not show a 2-hour discretized GIT structure. The path-based and optimized GIT-based solutions outperform the GIT-based solutions by approximately 3%, and between them remain similar, less than 1% of gap performance difference.

In the case of the large size set of instances, the path-based solutions do not show a 2-hour discretized GIT structure neither for the medium nor for the large truck type variants. However, they show more than 90% of a 2 and 4-hour discretized GIT structure. The path-based and the optimized GIT-based approaches strongly outperform the GIT-based approach by more than 10% of gap performance. This reveals that the GIT generation heuristic procedure for the GIT-based approach is not effective in producing high-quality discretized GITs. Interestingly, when enforcing an in-tree structure to the optimized GIT-based approach, the gap performance difference is 3.44% higher compared to the solutions of the path-based approach for the All-hubs instance.

Enforcing a discretized GIT structure has useful operational benefits, such as to enhance operational realism and an operationally feasible seamless plan which allows only the examination of the destination of the commodity and the bucket that the remaining available time of the commodity falls into, in order to determine the appropriate outbound trailer for loading. Nevertheless, this structure bears a penalty cost of around 2% of deterioration in the gap performance for a 2-hour discretized GIT, and around 3%-4% for more rigid GIT structures when compared to the solutions of the relaxation thereof for the All-hubs instance.

The next future avenue for research is to investigate alternative approaches for finding discretized GIT structures, test the approaches in other realistic instances to measure the penalty cost of imposing a GIT structure, and find tighter lower bounds of the problem for the large problems.

Appendices

APPENDIX A
OPERATIONS DESIGN FOR HIGH-VELOCITY INTRA-CITY PACKAGE SERVICE

A.1 Detailed algorithm to add cycles to the BP

To formally define this procedure, let $\mathcal{C}$ be the set of all cycles for a given regime. We define the set of late commodities as $\mathcal{K}^l$. A commodity is said to be late if it has at least one late arrival interval. For each late commodity k, we identify all late arrival intervals in the set $\mathcal{I}_k$. Let $\mathcal{A}_k^i$ be the set of arcs of the path of a late commodity k associated to the late arrival interval i. We also identify the parameter $wt_k^{(u,v),i}$ as the waiting time on each arc (u,v) of the path of a late commodity k in late arrival interval i. Similarly, we identify the parameter wt_k^{max} as the maximum allowable waiting time on each arc of the path of commodity k such that the commodity is on-time. Let T^s denote the set of different starting times allowed for cycles. Let w_k be the fraction of the arrival rate of commodity k over all commodity arrival rates. Let o^k be the on-time metric for commodity k. Finally, we define the parameter o^{target} as the overall on-time metric to achieve, and o^{min} as the minimum on-time metric that must be satisfied for all commodities.

```
1: while $o^{worst} < o^{min}$ and $o < o^{target}$ do
2:     if $o^{worst} < o^{min}$ then
3:         Identify the commodity $k$ having the minimum on-time metric, $o^{worst}$, call it $k^*$
4:         for $i \in \mathcal{I}_{k^*}$ do
5:             for $(u,v) \in \mathcal{A}_{k^*}^i$ do
6:                 if $wt_k^{(u,v),i} > wt_k^{max}$ then
7:                     $S(u,v) \leftarrow S(u,v) + w_k[wt_k^{(u,v),i} - wt_k^{max}]$
```

```
 8:             end if
 9:         end for
10:     end for
11: end if
12: if $o < o^{target}$ and $o^{worst} \geq o^{min}$ then
13:     for $k \in \mathcal{K}^l$ do
14:         for $i \in \mathcal{I}_k$ do
15:             for $(u, v) \in \mathcal{A}_k^i$ do
16:                 if $wt_k^{(u,v),i} > wt_k^{max}$ then
17:                     $S(u, v) \leftarrow S(u, v) + w_k[wt_k^{(u,v),i} - wt_k^{max}]$
18:                 end if
19:             end for
20:         end for
21:     end for
22: end if
23: Select $(u^*, v^*) = \underset{(u,v) \notin Tabu}{\text{argmax}} [s(u, v) + s(v, u)]$
24: if $(u^*, v^*) \neq (u_l^*, v_l^*)$ or $S_l \neq S(u^*, v^*) + S(v^*, u^*)$ then
25:     for $t \in T^s$ do
26:         if $o_{c^{(u^*, v^*, t)}} > o$ then
27:             $c \leftarrow c^{(u^*, v^*, t)}$
28:             $o \leftarrow o_{c^{(u^*, v^*, t)}}$
29:         end if
30:     end for
31:     Add cycle $c$ to the network
32:     Update $\mathcal{K}^l, \mathcal{I}_k, \mathcal{A}_k^i, o, o^{worst}$
33: else
34:     Add $(u^*, v^*)$ to $Tabu$
```

```
35:     end if
36:     $S_l \leftarrow S(u^*, v^*) + S(v^*, u^*)$
37:     $(u_l^*, v_l^*) \leftarrow (u^*, v^*)$
38: end while
```

Algorithm A.1 works as follows. We first aim to improve the minimum on-time metric for all commodities and subsequently we improve the overall on-time metric. In the first case, we identify the commodity having the minimum on-time metric, then we focus on improving the on-time metric for only that commodity (line 3). Next, we identify each late arrival interval and each arc of the commodity path (lines 4 and 5). Then, we check whether the waiting time on each arc of the commodity path is greater than the maximum waiting time allowed. If that is the case, we add the difference to the lateness score of the corresponding arcs weighted by the commodity arrival rate (lines 6 and 7). When we improve the overall on-time metric, we proceed in a similar fashion but focusing on all late commodities. We identify each late commodity, each late arrival interval, and each arc of the commodity path (from lines 13 - 15). Next, we verify whether the waiting time on each arc of the commodity path is greater than the maximum waiting time allowed. If this is the case, we add the difference to the lateness score of the corresponding arcs weighted by the commodity arrival rate (lines 16 and 17). After lateness scores are computed for each arc, we greedily select the arc with the highest lateness score (line 23). We verify that the arc selected is different than the previously chosen arcs or the lateness score is different than the previous iteration (line 24). We then evaluate different starting times of the cycle every t minutes on the time-space network and choose the one that returns the best on-time metric, which can be the minimum or the overall metric in consideration (lines 26 - 28). Finally, we add the cycle to the network and we update the late commodities and late arrival intervals (lines 31 - 32). In the case we obtain the same selected arc with the same lateness score value compared to the previous iteration, we do not add the cycle and we insert the arc to a tabu list for s iterations. We repeat the procedure until a minimum on-time metric is achieved for each commodity and a target overall on-time metric is reached.

A.2 Detailed Algorithm to Remove Cycles from the BP

Algorithm 6 Removal Cycle

```
1:  while Iterate do
2:      for $c \in \mathcal{C}$ do
3:          Compute $o^{worst,c}$, $o^c$, $u^{max,c}$ when cycle $c$ is removed
4:          if $o^{worst,c} > o^{min}$ and $o^c > o^{target}$ and $u^{max,c} \leq 100\%$ then
5:              Compute the expected return of removing cycle $c$, $r^c$
6:              if $r^c < 0$ and $r^c < r^{min}$ then
7:                  $c^* \leftarrow c$
8:                  $r^{min} \leftarrow r^c$
9:                  Iterate $\leftarrow True$
10:             end if
11:         end if
12:     end for
13:     $\mathcal{C} \leftarrow \mathcal{C} \setminus c^*$
14: end while
```

The cycle removal algorithm works as follows: In each iteration, we choose the cycle that has the lowest expected return for removal. To do so, we first verify whether the cycle is admissible for removal, i.e., the minimum (worst) on-time metric, minimum overall on-time metric, and maximum utilization (computed on the flat network) are not violated when the cycle is removed (lines 3 and 4). Next, among the cycles that are admissible for removal, we identify the lowest expected return of a cycle according to the return of removing a cycle defined in 2.17 (lines 5 - 9). Finally, we remove the cycle and keep iterating in a similar way until there is no longer an admissible cycle or there is no negative return of removing a cycle.

APPENDIX B
SERVICE NETWORK DESIGN WITH HUB SELECTION

B.1 Initial Experiments Using Large values of s

In these experiments, we group time requirements of commodities according to the following time requirement groups:

Table B.1: Time Requirement Groups

Range (Hrs)	Time Requirement Set (hours)
(0, 4]	4
(4, 8]	4
(8, 12]	8
(12, 16]	12
(16, 20]	16
(20, 30]	20
(30, 40]	30
>40	40

After we group commodities according to the time requirement groups defined in Table B.1, we obtain 6681 commodities, and the time requirement of half of them, according to the time requirement group they belong, is not sufficient to cover at least two times the direct travel time route. We redefine the time requirement for those commodities to have at least two times the direct travel time. Since the truck cost per kilometer for arcs is not available, we compute an average arc truck cost per kilometer from the available truck arc cost. Given that the capacity, in terms of packages, of hubs is not available, we consider it as unrestricted. We employ mainly two truck sizes: 7T and 14T, which depends on the arc distance. We only allow paths having at most two intermediate stops and the length of paths is restricted by the time requirement of each commodity. We experiment with a different maximum number of outbound destinations, L, and a maximum number of intermediate hubs, s.

In Figures B.1, B.2, B.3 and B.4, we illustrate how the number of intermediate stops for commodity paths and the cost of the solution change as we vary s and L. We fix L as 5 times the actual l_u for each $u \in \hat{\mathcal{U}}$ for Figures B.1, B.2, and we fix $s = 20$ for Figures B.3 and B.4.

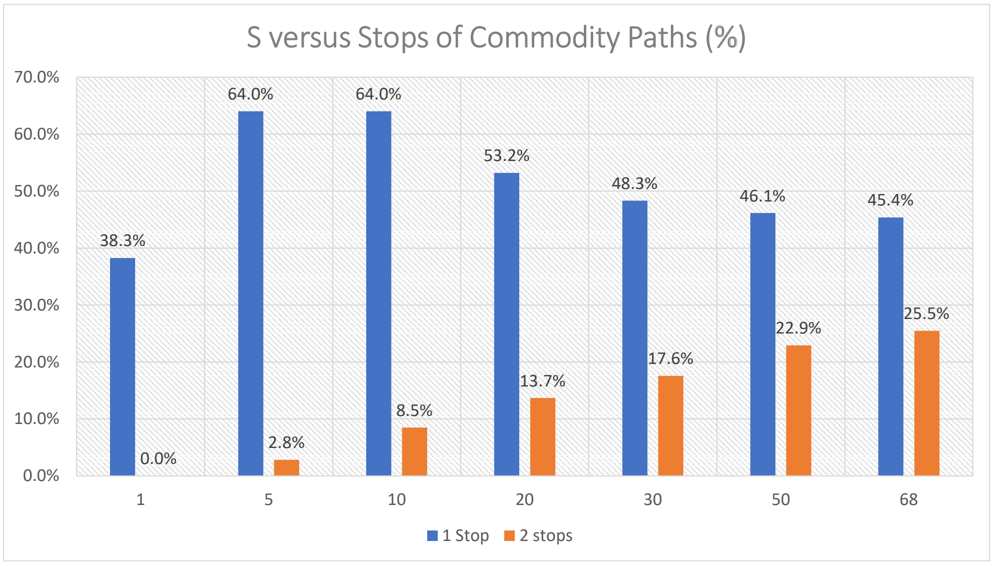

Figure B.1: s versus Commodity Stops

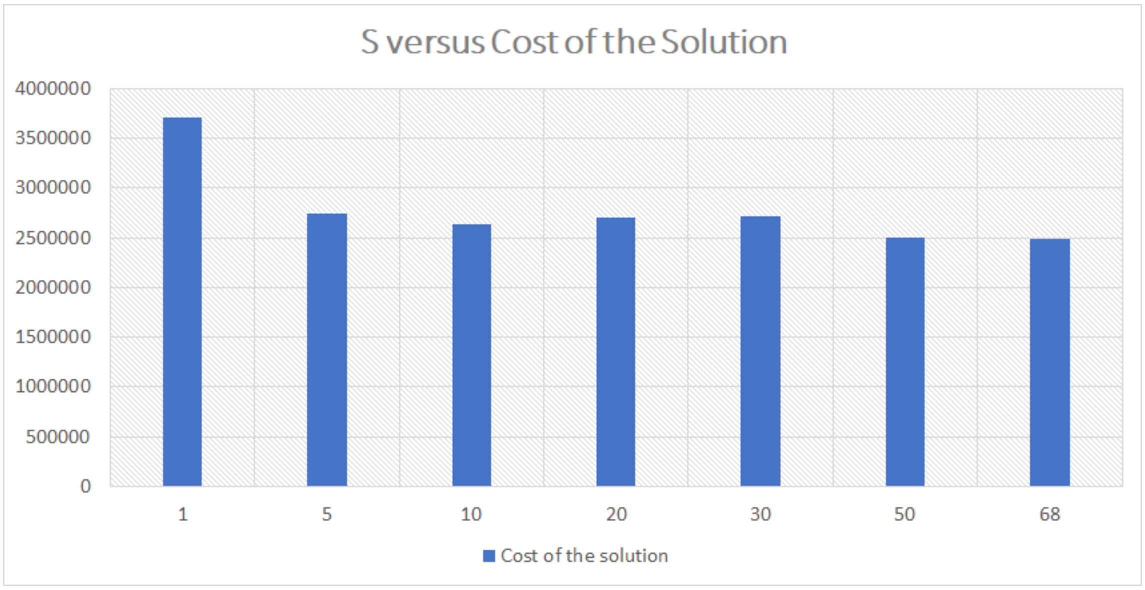

Figure B.2: s versus Solution Cost

In Figure B.1, we observe that the percentage of commodities taking non-direct paths increases significantly for small s, however, for larger s (such as $s \geq 30$) the increase is not significant; in fact we see more commodity paths with 2 stops and fewer with 1 stop. In Figure B.2, we observe that there is a huge improvement in the cost of the solution from $s = 1$ to $s = 5$, nearly 30%. Nevertheless, from $s = 5$ onward, we do not notice a significant improvement. This evidence strongly suggests that in practice, a large number of intermediate hubs is not required to obtain good solution costs.

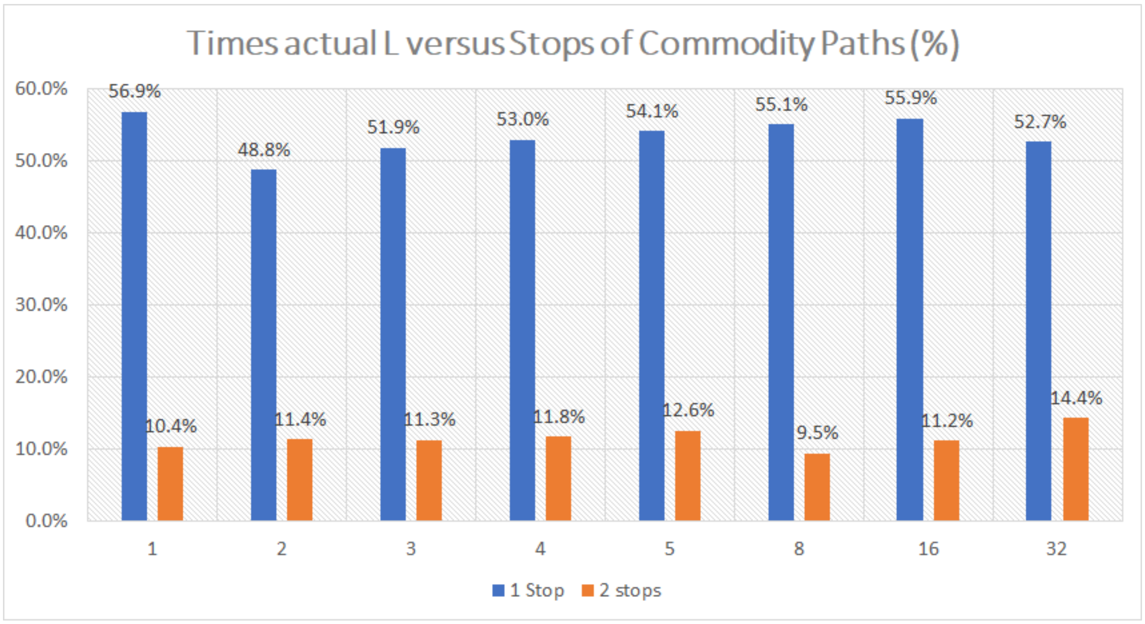

Figure B.3: L versus Commodity Stops

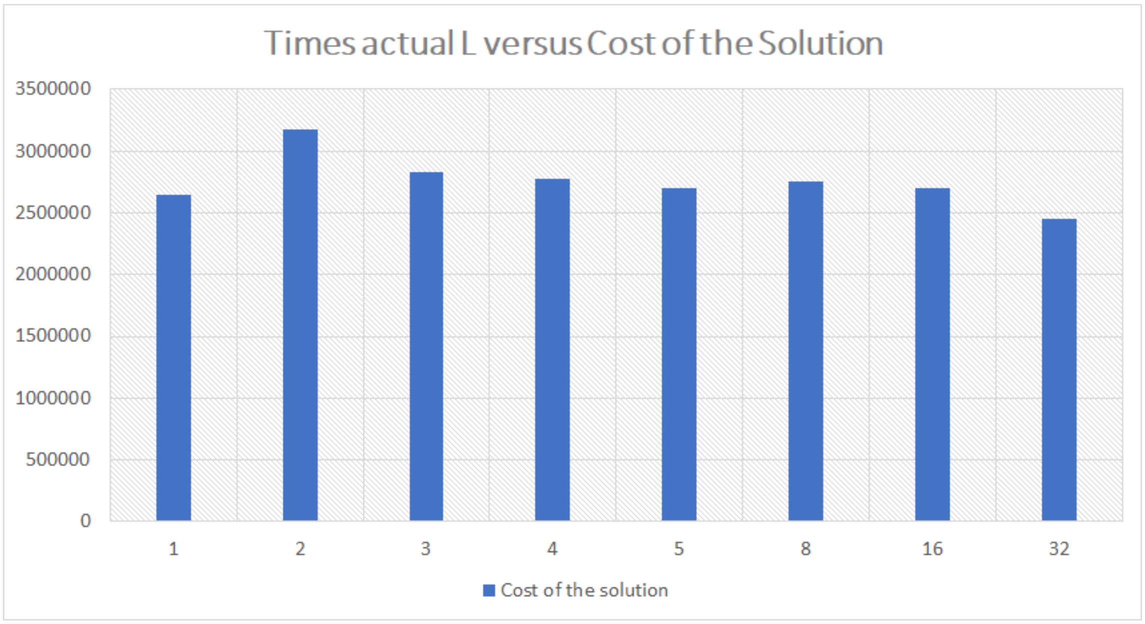

Figure B.4: L versus Solution Cost

In Figure B.3 and B.4, we do not notice a significant difference in the number of stops for commodity paths and the cost of the solution across different values of L, which suggests that this restriction may not be of importance, except to some hubs that currently have few loading docks.

REFERENCES

[1] T. Deloison, E. Hannon, A. Huber, B. Heid, C. Klink, R. Sahay, and C. Wolff, “The future of the last–mile ecosystem,” *Ecosystem. World Economic Forum*, 2020.

[2] P. B. Inc., “Pitney bowes parcel shipping index reports continued growth as global parcel volume exceeds 100 billion for first time ever,” 2020.

[3] T. Crainic, “Service network design in freight transportation,” *European Journal of Operational Research*, vol. 122, pp. 272–288, 2000.

[4] N. Wieberneit, “Service network design for freight transportation: A review,” *OR Spectrum, Springer*, vol. 30, pp. 77–112, 2008.

[5] C. Barnhart, C. A. Hane, and P. H. Vance, “Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems,” *Operations Research*, vol. 48, no. 2, pp. 318–326, 2000.

[6] J. Andersen, M. Christiansen, T. G. Crainic, and R. Grønhaug, “Branch and price for service network design with asset management constraint,” *Transportation Science*, vol. 45, no. 1, pp. 33–49, 2011.

[7] B. Gendron and M. Larose, “Branch-and-price-and-cut for large-scale multicommodity capacitated fixed-charge network design,” *EURO Journal on Computational Optimization*, vol. 2, pp. 55–75, 2014.

[8] X. Li, Y. P. Aneja, and J. Huo, “Using branch-and-price approach to solve the directed network design problem with relays,” *Omega*, vol. 40, pp. 672–679, 2012.

[9] J. M. Farvolden and W. B. Powell, “Subgradient methods for the service network design problem,” *Transportation Science*, vol. 28, no. 3, pp. 177–272, 1994.

[10] A. Armacost, C. Barnhart, and K. Ware, “Composite variable formulations for express shipment service network design,” *Transportation Science*, vol. 36, no. 1, pp. 1–20, 2002.

[11] C. Barnhart and R. R. Schneur, “Air network design for express shipment service,” *Operations Research*, vol. 44, no. 6, pp. 852–863, 1996.

[12] A. Armacost, C. Barnhart, K. Ware, and A. Wilson, “Ups optimizes its air network,” *INFORMS Journal on Applied Analytics*, vol. 34, no. 1, pp. 15–25, 2004.

[13] C. Barnhart, N. Krishnan, D. Kim, and K. Ware, "Network design for express shipment delivery," *Computational Optimization and Applications*, vol. 21, no. 3, pp. 239–262, 2002.

[14] W. Powell and Y. Sheffi, "The load planning problem of motor carriers: Problem description and a proposed solution approach," *Transportation Research Part A*, vol. 17, no. 6, pp. 471–480, 1983.

[15] W. Powell, "A local improvement heuristic for the design of less-than-truckload motor carrier networks," *Transportation Science*, vol. 20, no. 4, pp. 227–291, 1986.

[16] W. Powell and Y. Sheffi, "Design and implementation of an interactive optimization system for network design in the motor carrier industry," *Operations Research*, vol. 37, pp. 12–29, 1989.

[17] K. O. Kyoung, C. D. Martland, and J. M. Sussman, "Routing and scheduling temporal and heterogeneous freight car traffic on rail networks," *Transportation Research Part E: Logistics and Transportation Review*, vol. 34, no. 2, pp. 101–115, 1998.

[18] M. Zhang, M. Janic, and L. Tavasszy, "A freight transport optimization model for integrated network, service, and policy design," *Transportation Research Part E: Logistics and Transportation Review*, vol. 77, pp. 61–76, 2015.

[19] L. Duan, L. A. Tavasszy, and J. Rezaei, "Freight service network design with heterogeneous preferences for transport time and reliability," *Transportation Research Part E*, vol. 124, pp. 1–12, 2019.

[20] T. Crainic, M. Hewitt, M. Toulouse, and D. Vu, "Service network design with resource constraints," *Transportation Science*, vol. 50, pp. 1380–1393, 2014.

[21] T. Crainic, N. Ricciardi, and G. Storchi, "Models for evaluating and planning city logistics systems," *Transportation Science*, vol. 43, no. 4, pp. 432–454, 2009.

[22] T. G. Crainic, N. Ricciardi, and G. Storchi, "Advanced freight transportation systems for congested urban areas," *Transportation Research Part C: Emerging Technologies*, vol. 12, no. 2, pp. 119–137, 2004.

[23] T. G. Crainic, F. Errico, W. Rei, and N. Ricciardi, "Modeling demand uncertainty in two-tier city logistics tactical planning," *Transportation Science*, vol. 50, pp. 363–761, 2016.

[24] A. Lium, T. G. Crainic, and S. W. Wallace, "A study of demand stochasticity in service network design," *Transportation Science*, vol. 43, pp. 144–157, 2009.

[25] T. Crainic and A. Sgalambro, "Service network design models for two-tier city logistics," *Optimization Letters, Springer*, vol. 8, pp. 1375–1387, 2014.

[26] D. Kim and C. Barnhart, "Transportation service network design: Models and algorithms," *Computer-Aided Transit Scheduling*, vol. 471, pp. 259–283, 1999.

[27] J. Andersen, T. G. Crainic, and M. Christiansen, "Service network design with asset management: Formulations and comparative analyses," *Transportation Research Part C*, vol. 17, no. 2, pp. 197–207, 2009.

[28] A. Erera, M. Hewitt, M. Savelsbergh, and Y. Zhang, "Improved load plan design through integer programming based local search," *Operations Research*, vol. 47, no. 3, pp. 412–427, 2013.

[29] A. I. Jarrah, E. Johnson, and L. C. Neubert, "Large-scale, less-than-truckload service network design," *Operations Research*, vol. 57, no. 3, pp. 609–625, 2009.

[30] M. Chouman and T. G. Crainic, "Cutting-plane matheuristic for service network design with design-balanced requirements," *Transportation Science*, vol. 49, no. 1, pp. 99–113, 2015.

[31] M. Savelsbergh and T. V. Woensel, "50th anniversary invited article?city logistics: Challenges and opportunities," *Transportation Science*, vol. 50, no. 2, pp. 579–590, 2016.

[32] G. Jia, R. Ma, and Z. Hu, "Review of urban transportation network design problems based on citespace," *Mathematical Problems in Engineering*, vol. 2019, 2019.

[33] R. Z. Farahani, E. Miandoabchi, W. Y. SzetoW, and H. Rashidi, "A review of urban transportation network design problems," *European Journal of Operational Research*, vol. 229, pp. 281–302, 2013.

[34] T. G. Crainic and J. M. Rousseau, "Multicommodity, multimode freight transportation: A general modeling and algorithmic framework for the service network design problem," *Transportation Research Part B: Methodological*, vol. 20, no. 3, pp. 225–242, 1986.

[35] D. Kim, C.Barnhart, K. Ware, and G. Reinhardt, "Multimodal express package delivery: A service network design application," *Transportation Science*, vol. 33, no. 4, pp. 391–407, 1999.

[36] J. Campbell, "A survey of network hub location," *Studies in Locational Analysis*, vol. 6, pp. 31–49, 1994.

[37] ——, "Hub location and the p-hub median problem," *Operations Research*, vol. 44, no. 6, pp. 1–13, 1996.

[38] J. F. Campbell, A. T. Ernst, and M. Krishnamoorthy, "Hub location problems," *Facility Location: Applications and Theory. Springer*, 2002.

[39] M. E. O'Kelly and H. J. Miller, "The hub network design problem: A review and synthesis," *Journal of Transport Geography, Elsevier*, vol. 2, pp. 31–40, 1994.

[40] M. Kuby and R. Gray, "Hub network design problem with stoppers and feeders: Case of federal express," *Transportation Research*, vol. 27, pp. 1–12, 1993.

[41] D. Kim and C. Barnhart, "Transportation service network design: Models and algorithms," *Computer-Aided Transit Scheduling*, vol. 471, pp. 259–283, 1999.

[42] S. Huber, J. Klauenberg, and C. Thaller, "5consideration of transport logistics hubs in freight transport demand models," *European Transport Research Review, Springer*, 2015.

[43] S. Erol and A.S.Duyguvar, "The evolution of logistics hubs and a conceptual framework for logistics hubs location decisions," *Using Decision Support Systems for Transportation Planning Efficiency*, vol. Chapter 5, 2016.

[44] S. Alumur, B. Kara, and O. Karasan, "Multimodal hub location and hub network design," *Omega*, vol. 40, no. 6, pp. 927–939, 2012.

[45] R. DeCamargo and G. Miranda, "Single allocation hub location problem under congestion: Network owner and user perspective," *Expert Systems with Applications*, vol. 39, no. 3, pp. 3385–3391, 2012.

[46] I. Rodríguez-Martín, J. Salazar-Gonzalez, and H. Yaman, "A branch-and-cut algorithm for the hub location and routing problem," *Computers Operations Research*, vol. 50, pp. 161–174, 2014.

[47] S. Gelareh and S. Nickel, "Hub location problems in transportation networks," *Transportation Research Part E, Logistics and Transportation Review*, vol. 47, no. 6, pp. 1092–1111, 2011.

[48] B. Luttrell, "Https://www.areadevelopment.com/logisticsinfrastructure/intermodal-sites-q1-2015/site-selection-process-supplychain-optimization-linked-74421.shtml," 2015.

[49] T. G. Crainic, X. Fu, M. Gendreau, W. Rei, and S. W. Wallace, "Hub location problems," *Progressive hedging-based metaheuristics for stochastic network design*, vol. 58, no. 2, pp. 114–124, 2011.

[50] C. Barnhart, E. Johnson, G. Nemhauser, M. Savelsbergh, and P. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Operations Research*, vol. 56, no. 3, pp. 316–329, 1998.

[51] A. Rothenbächer, M. Drexl, and S. Irnich, "Branch-and-price-and-cut for a service network design and hub location problem," *European Journal of Operational Research*, vol. 255, pp. 935–947, 2016.

[52] B. Yıldız, O. E. Kara, and H. Yaman, "Branch-and-price approaches for the network design problem with relays," *Computers and Operations Research*, vol. 92, pp. 155–169, 2018.

[53] T. Capelle, C. E. Cortes, M. Gendreau, P. A. Rey, and L. Rousseau, "A column generation approach for location-routing problems with pickup and delivery," *European Journal of Operational Research*, vol. 272, no. 1, pp. 121–131, 2019.

[54] C. Contardo, J. Cordeau, and B. Gendron, "An exact algorithm based on cut-and-column generation for the capacitated location-routing problem," *INFORMS Journal on Computing*, vol. 26, no. 1, pp. 88–102, 2014.

[55] A. Raith, "Speed-up of labelling algorithms for biobjective shortest path problems," *Proceedings of the 45th Annual Conference of the ORSNZ*, 2010.

[56] D. Duque, L. Lozano, and A. L. Medaglia, "An exact method for the biobjective shortest path problem for large-scale road networks," *European Journal of Operational Research*, vol. 242, no. 3, pp. 788–797, 2015.

[57] A. Giret, Y. Kergosien, G. Sauvanet, and E. Neron, "An efficient label-setting algorithm for the bi-objective shortest path problem," *In Proceedings of 5th the International Conference on Operations Research and Enterprise Systems*, pp. 197–203, 2016.

[58] A. Sedeño-noda and M. Colebrook, "A biobjective dijkstra algorithm," *European Journal of Operational Research*, vol. 276, no. 1, pp. 106–118, 2019.