Improving supply chain performance

Doctoral thesis

IMPROVING SUPPLY CHAIN PERFORMANCE

ORDER PICKING AND SERVICE NETWORK DESIGN

Farzaneh Rajabighamchi

2023

IMPROVING SUPPLY CHAIN PERFORMANCE

ORDER PICKING AND SERVICE NETWORK DESIGN

Dissertation

To obtain the degree of Doctor at Maastricht University,
on the authority of the Rector Magnificus, Prof. dr. Pamela Habibović,
in accordance with the decision of the Board of Deans,
to be defended in public
on Tuesday 5 of March 2024, at 16.00 hours

by

Farzaneh Rajabighamchi

Promotor
Prof. dr. Stan van Hoesel (Maastricht University)
Prof. dr. Inneke Van Nieuwenhuyse (Hasselt University)

Copromotor
dr. Christof Defryn (Maastricht University)
Prof. dr. Kris Braekers (Hasselt University)

Assessment Committee
Prof. dr. Célia Paquay (Université de Liège)
Prof. dr. Kenneth Sörensen (University of Antwerp)
Prof. dr. Fred Vermolen (Hasselt University)
Prof. dr. Tjark Vredeveld (Chair, Maastricht University)

Cover
Production Publisher Maastricht University
ISBN 978-94-6469-386-7

To my Family, for their unconditional love and eternal support

Acknowledgments

"If I have seen a little further it is by standing on the shoulders of giants" – Sir Isaac Newton (1676). There are so many people to whom I feel the need to express my deepest gratitude, as they have raised me up and allowed me to reach beyond. First, I would like start by thanking my supervisors Prof. Stan van Hoesel and Dr. Christof Defryn for their invaluable advice and continuous support during my PhD study. Their immense knowledge and experience have encouraged me in all the time of my academic research and daily life. Very special thanks to Stan who was always approachable and advising me to enjoy my life as much as possible. There was never a moment in which I felt alone, and I knew I could always count on a word of understanding, motivation and orientation from you, even when the path was a bit cloudy. I am incredibly grateful for all the help and support my supervisor in Hasselt University, Inneke van Nieuwenhuyse gave me. She was always patient and helpful with my bothering questions. I would also like to thank all the members of the assessment committee for taking the time to carefully read the thesis and for their invaluable comments.

I would like to express my highest love and gratitude to my lovely family. Without their eternal love, tremendous unconditional support and encouragement in my life, it would be impossible for me to complete my study. There is no bigger treasure than having you all in my life and no bigger joy than to be able to share this achievement with you. Thanks in believing in me and teaching me to be the best version of myself. Your sacrifices and tireless dedication to my well-being have shaped me into the person I am today. Your guidance has been my compass, navigating me through life's challenges with a warm heart. I would like to especially thank my sister, whom without this would have not been possible. Reyhaneh, your support, understanding, and love mean the world to me. Through thick and thin, your presence has been my constant source of strength. You have always been there for me whenever I needed. You are not only my sister, but also my best friend in life, my partner in crime, and my rock. I have learned a lot from you and I am very thankful for everything.

I am especially grateful to my best friend Ebrahim for all his helps and support. Ebrahim, for being there at every step of the way. Thank you for bringing me up, for calming me down, for making me feel confident and capable.

It would have been very hard without you. I am very proud and grateful for having you by my side.

I am very thankful for my Iranian colleagues, Ashkan and Arman. Spending time together with our Iranian colleagues in the Netherlands has been a source of comfort and camaraderie, especially as we navigate the challenges of being away from our homeland. The shared language and culture create a unique bond that goes beyond professional collaboration. Whether it's sharing stories over a cup of Persian tea, celebrating cultural festivities, or simply supporting each other during moments of homesickness, the sense of community we've cultivated is invaluable. In these shared moments, we find solace and connection, making our workplace feel like a home away from home.

I would like to thank my friend and colleague, Son Tran for a cherished time spent together in the KE. Also my sincere thanks to my colleagues Li, Caterina, Luca, Niels, Dewi, Aida, Francois, Freija, Marie, Joep, Francesco, Niloofar and Daniel for making PhD life very exciting and happy for me. I would like to express my sincere gratitude to my amazing friends Shafagh and Sasan for always being there for me. Knowing that you are by my side, made it more bearable for me to go through this adventure. Shafagh, you are like a sister to me and your company makes me very happy and delighted. I am deeply grateful to Johaness, who have been like an older brother for me and I am very thankful to have him in my life.

Many thanks to my dearest friends and colleagues Aditya and Benoit who made the first two years of my PhD , the best years of my life. Benoit, I deeply appreciate your kindness and hospitality that made me feel at ease and that I can count on you as a great friend of mine. Aditya, although you were not the best neighbour :D, but you were a kind, fun, and easygoing colleague with whom I had the happiest time.

I extend my deepest appreciation to my devoted boyfriend, Mathijs, whose unwavering support, encouragement, and understanding have been the pillars of my academic journey. His patience, motivation, and belief in my abilities have been instrumental in overcoming the challenges and obstacles that inevitably arise during such an extensive research undertaking. I am also profoundly thankful to his family for being there for me and for their hospitality, love and support, that allowed me to focus on my work with a sense of comfort and belonging.

Farzaneh Rajabighamchi

Contents

List of Figures

List of Tables

1 Introduction

Supply chain optimization is a widely studied topic in Operations Research with work on topics from production and service planning, storing and inventory management, and scheduling, routing, sourcing, and distribution among others.

In this thesis, two different supply chain problems are investigated. This introductory chapter establishes important concepts underlying the results in this thesis. In Section 1.1, the notion of order picker routing problem is formally introduced and two different variants of this problem which includes chapters two and three of this thesis, are discussed. In Section 1.2, we introduce the multi-commodity network design problem addressed in the fourth chapter.

1.1 Order picker routing in the warehouse

Warehouses play a crucial role in the supply chain management since they are used by all different types of businesses that need to temporarily store products in bulk before either shipping them to other locations or individually to end consumers. Warehousing makes it easier to receive, store and distribute the goods as all the goods are stored in a central location. This helps in reducing transportation costs and increasing the value of goods as products are available at the right place, at the right time. One of the processes within warehouses that provide significant cost-saving potential is the so-called order picking.

Valle, Beasley, and Cunha [168] define order picking as retrieving products from storage in response to specific customer requests and it is mostly done manually by dedicated employees called pickers. Despite numerous attempts to automate the picking process, manual order picking methods are still widely used in industry [46]. Picker-to-part order picking systems are the most significant manual methods [88]. In such systems, order pickers fulfill customer orders by traversing the picking area of the warehouse. Picking orders are customer orders that are processed on the same tour. Every tour begins and ends at the depot, and it covers all of the storage locations (pick locations) for the required items that are included in the corresponding picking order.

According to Tompkins et al. [161], this process can account for as much as 55% of the total warehouse operating cost. The order picking process comprises various activities such as setup, searching, and traveling between items. Tompkins et al. [161] state that the latter can consume 50% of an order-pickers time, which in manual-picking systems constitutes high costs. Hence, optimizing the routes taken by order-pickers provides significant savings potential. In addition, shorter traveling distance results in a quicker picking process, which is a crucial link between order picking and service levels. Order picking is part of the process of fulfilling customer orders. The speed and efficiency of the order picking process can affect the delivery time of a customer's order, which can affect customer satisfaction.

A tour's length is influenced by the sequence in which the selected locations are meant to be visited. The Picker Routing Problem includes determination of this sequence as well as the shortest route to collect them and it can be phrased as follows. Let a list of picking orders be given which includes the required products and their known storage locations. The sequence in which the pick locations are to be visited for each picking order, as well as the path across the warehouse's picking area corresponding to it, must be decided in order to minimize the length of the overall tour.

The Picker Routing Problem has been extensively researched in the literature, and there are numerous different techniques to solve it. However, the majority of methods apply simple routing techniques such as S-shape strategy, which is prevalent due to its practicability and simplicity of implementation, that could lead to quite protracted journeys [136]. These routing techniques used in reality result in trips that are up to 48% longer than an optimal tour [160]. Given that the development of such long tours is likely to have a major negative influence on the efficiency of the picking process, the approaches offered to date cannot be deemed satisfactory.

In the second and third chapter of this thesis, we focus on the Order picker routing problem in the warehouse- Therefore, It is very important to describe the warehouse layout addressed in this thesis and further supplementary information on storage and picker routing.

Chapter 2 of this thesis deals with the single order picker routing problem within a conventional multi-block warehouse layout. This problem entails collecting items from storage in response to customer requests. We consider the case where every product in the warehouse has a unique location and the order picker must visit only that specific location in order to pick the product in the order list.

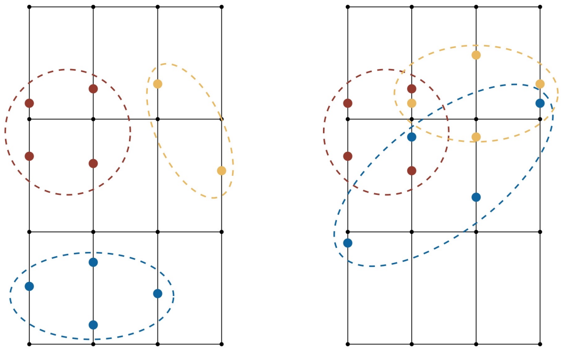
In the literature, exact algorithms only exist for small warehouses with few cross aisles, while for other larger warehouse types some heuristic and meta-heuristic methods are provided.

This chapter presents a new efficient exact pre-processing algorithm for graph reduction, removing the unnecessary vertices and edges from the graph which are not part of the optimum solution, resulting in a significant reduction in the calculation time. Though most warehouse lay-outs in the literature are indeed modelled on grid graphs, this is not a limitation for our algorithm. The proposed algorithm can be used for any warehouse layout, as long as it is represented as a planar graph., which makes this method more practical. The presented method allows us to solve adequately big (more realistic) instances in few seconds and it gives the optimum solution for all the instances.

Ho and Sarma [74] state that while items of the same stock keeping unit (SKU) are generally co-located in traditional warehouses, tracking technologies and automatic identification allow for free-form storage (scattered storage policy). This implies that identical SKUs can be positioned in multiple locations throughout the warehouse. In **chapter 3**, we will answer the following two questions: Which SKU locations should be visited to pick the requested items? What is the shortest route containing the chosen locations? This extended problem can be formulated as a Generalized Traveling Salesman Problem (GTSP), where a cluster is defined as containing all identical SKUs.

In the last decades, there have been several variants of the TSP studied including generalized traveling salesman problem (GTSP) in which the graph nodes are partitioned into a group of clusters. The objective is to find a minimum-cost tour spanning a subset of items such that exactly one item from each cluster is included in the tour. Due to its various real-world practical applications, the GTSP has drawn considerable attention in mathematical models, approximation and heuristic and meta-heuristic algorithms. One of the applications for the GTSP is the order picking problem in the warehouse. Each cluster represents possible locations associated with a specific product, and it is assumed that a single order picker is able to deliver all the products on the same tour.

While the storage assignment of items in the warehouse constitutes a complex problem, it is not unlikely to assume that clusters will often overlap. El Krari et al. [54] define overlapping clusters as those that share a geographical space, i.e. when drawing their borders, they find themselves intertwined. Figure 1.1 shows examples of non-overlapping and overlapping clusters.



(a) Non-Overlapping (b) Overlapping

Figure 1.1: *Different clustering topologies.*

In this dissertation, I propose an MILP formulation of the problem and present a heuristic solution method for the GTSP with geographical overlap between clusters, as all products are scattered throughout the warehouse and the product clusters are not separated geographically, which is a more general form of GTSP. The motivation for this research stems not only from the potential benefits that can be achieved within the efficiency of a warehouse but, also because of the fact that very little attention has been given to this problem when accounting for the overlap of clusters. To the best of our knowledge, there exists no paper in the literature taking this aspect of the problem (overlapping clusters in the order picking problem) into account. Our proposed exact model for GTSP, outperforms the existing exact models for GTSP in the literature (with different applications than the order picking problem). We propose a Guided Local Search (GLS) algorithm that exploits problem-specific information during the search procedure to guide the local search operators to promising areas of the search space. The algorithm is implemented on a set of benchmark instances from the literature.

1.2 Multi-commodity network design problem

Over the last decades, technological advances and economic development have drastically changed numerous facets of our society. Higher standards of living have not only led to increased demand for goods, but also to a shift in preferences in the customers' decision-making process. Convenience has become a key factor driving customers' purchase decisions. E-commerce is thriving and home delivery has evolved to a must for a large group of customers, in order to benefit from faster or customizable delivery options. To stay competitive in this environment, businesses have to adapt their strategies to these changes in consumer behavior.

Courier service companies, such as Amazon, Walmart, Ebay Gati, and Blue Dart have recognized customer thirst for convenience and seized this opportunity, offering a wide range of on-demand services. These couriers collect historic data on the deliveries that were requested by repeat customers, as well as one-time customers, and they can utilize these data to make a reasonable forecast for future demands. Since the transportation of goods is not a value adding process and transportation makes up a significant part of the operational costs, it should come to no surprise that efficient route planning lies at the core of their functioning. Couriers try to minimize their total costs. One of the most efficient ways to reduce the total costs is by incorporating express services into their route planning which allows for cutting costs by outsourcing (if they can not do it themselves) inconvenient requests, and over-demand.

In today's international context, the planning and coordination of all necessary logistics operations within a supply network is a tedious task. As supply chain partners often established strong dependencies on each other — with the aim to improve overall efficiency of the network —, any delay or disruption in the transport flows between these partners will create a significant impact on the underlying operations [4, 52, 37].

To plan and execute all required logistical operations within the supply chain, companies rely on third party logistics service providers (3PLs). These 3PLs manage the flow of goods between the different supply chain entities by either dispatching their own vehicles or by subcontracting logistics service providers to execute the required transportation requests [140, 118, 94]. A 3PL is responsible for coordinating all material flows that belong to the supply network. The network consists of multiple hubs, which either take the form of transshipment points within the supply network or represent a local supply node uniting multiple suppliers within a certain region. The flow density between each pair of hubs varies significantly over time (some connections are seldom used, others have high volumes every day) and is uncertain (exact volumes are only known last-minute). As the 3PL does not have its own fleet of trucks, it relies on — often local — subcontractors (carriers) to execute the transports.

We distinguish two types of agreements between the 3PL and its subcontractors. First, there is a long-term agreement to establish a periodic, fixed capacity on some of the network connections. For example, a truck is chartered every Monday and Thursday on a predefined link. As these long-term commitments are valuable to the carriers, competitive prices can be negotiated for the service. However, setting up this collaboration comes with a cost mainly related to Loss of control, sharing resources, integration of information and technology, trust, or willingness to change, etc. On top of that, sufficient flow should be guaranteed as the payment is always for the full truckload (i.e., independent from the actual load). Second, the 3PL can book an ad-hoc express delivery on the spot market. This service is more flexible and its cost depends on the volume and trajectory of the actual load.

Shipping operations must be well coordinated in order to meet the due dates of customer orders. This gives rise to a multi-commodity network design problem with delivery time, namely how the commodities of the various orders are to be assigned and in which sequence and in which point of time the hubs are to be visited.

We focus on the network design problem with split demand, where we allow the commodities to be split over multiple paths from their origins to their destinations. Furthermore, we consider a case in which the logistic companies make use of two distinct modes of transportation in order to transport the commodities between hubs: scheduled trucks (services), which operate on a fixed schedule for each period (i.e, one week), and this schedule is repeated periodically throughout the planning horizon (year or season etc.), and express delivery, which is used to meet demand that exceeds scheduled deliveries (services) or that requires expedited delivery. While a scheduled vehicle is strictly less expensive than an express vehicle, it must be paid regardless of capacity utilization, whereas the express truck's cost is determined by the package's weight and total distance traveled. This further complicates the problem by mandating the formation of a cost-effective equilibrium between scheduled and express vehicles capable of delivering all commodities within the specified time periods.

Furthermore, we considered the finite inventory capacity at each hub, which to the best of our knowledge is rarely taken into account in the existing literature. the amount of capacity used in each hub depends on the transport decisions taken, and varies over time; thus, it must be taken into account as a constraint for future transport decisions. The amount of capacity used in each hub on a given day has an impact on the capacity available the next day, implying that we must account for variations in hub capacities (in literature, each day has an independent capacity from previous or following days and in the day after, everything starts again to the default capacity). Moreover, we assume the overall demand for the commodity served by each route cannot exceed the vehicle capacity.

We consider uncertain demand for each period, therefore, we make use of a scenario-based two-stage stochastic model to solve the problem. Monte Carlo sampling and Sample Average Approximation (SAA) Method is used for constructing scenarios. Additionally, we consider a delivery date for each commodity to its destination hub. The cost of inventory storage may differ in different hubs, therefore the inventory cost must be included in our objective function and is added to the travel cost for the associated commodity to get the total cost. Additionally, we have no limits on vehicles returning to their origin at the end of the time horizon. We are just concerned with the number of vehicles at each hub remaining constant at the end of the period. In other words, the number of available vehicles in each hub at the start of each time horizon (beginning of the week) should be equal to the number at the end.

Chapter 4 of this thesis aims to cover several logistical decisions such as transport mode selection, freight planning and load allocation to these modes, fleet sizing, service routing and scheduling, and more importantly the inventory management in the hubs. We focus on the network design problem with split demand, where we allow the commodities to be split over multiple routes from their origins to their destinations.

We present an MILP formulation for the multi-commodity service network design problem with delivery time that includes express deliveries and uncertainty in demand.

The objective is to reduce total costs throughout the planning horizon. Due to the NP-hardness of the problem at hand, we employ a column generation strategy, in which the pricing problem is recast as a routing problem for each commodity, embedded in a branch-and-bound framework.

The main contributions of the chapter are the following: first, we consider multi-commodity network design models over time to allow differentiation between the (periodic) scheduled truck services and the ad-hoc express delivery option. Second, we account for potential capacity limitations in the hub and manage inventory levels accordingly. Furthermore, we incorporate the delivery time for commodity delivery, which makes our model more realistic. We develop competitive solution approaches based on an integration of column generation and branch-and-price to solve this realistic variant of the Multi-Commodity Network Design Problem (MCNDP). Moreover, we extend our models and results to a setting with uncertain demand and present a two-stage, scenario-based stochastic model which is solved using the average approximation method. Finally, a broad range of managerial insights have been generated by means of an extensive sensitivity analysis.

2

Graph reduction for the planar Travelling Salesman Problem. An application in order picking

Adapted from: [. Rajabighamchi, van Hoesel, and Defryn [132]]

Abstract

This chapter presents an improved exact algorithm for solving the order picker routing problem, a special case of the planar Travelling Salesperson Problem. The algorithm heavily relies on graph reduction techniques: it removes unnecessary vertices and edges from the planar graph that are not necessary in the optimal solution. As a result, we achieve a significant increase in calculation speed and reduction in the running time. The order picker routing problem entails collecting items from storage in response to customer requests. We use the Traveling Salesperson Problem (TSP) to optimize the routes taken by order pickers. In the literature, exact algorithms –typically based on dynamic programming- only exist for small warehouses with a small number of blocks, while for larger warehouse layouts mainly heuristic and meta-heuristic methods are provided.

The presented graph reduction method allows us to solve larger — more realistic — instances in a short amount of time. Our algorithm is tested on different problem instances from the literature and its performance is compared with the current state-of-the-art. We conclude that our algorithm outperforms existing algorithms in terms of simplicity, size and calculation time.

2.1 Introduction

For businesses operating with physical products, warehouses play an integral part in the efficiency of their supply chains. [64] highlight that — while modern supply chains have aimed at reducing inventory through initiatives such as "Just-In-Time" — warehouses are still present in most supply chain stages. The warehousing service is a very important component of the logistics system and plays a vital role in the supply chain process by balancing supply and demand. Thanks to the rapid growth of the e-commerce sector (accelerated by the COVID19 pandemic), the number of warehouses has even increased considerably over the last decade [50, 29]. As a result, supply chain sectors and specifically warehouses are forced to further streamline processes by increasing efficiency and cutting costs, while still ensuring high service levels to their customers [Dembińska [49] and Gutelius, Theodore, et al. [65]].

A warehouse process that provides significant cost-saving potential, is the *order picking process* as it is estimated to account for up to 55 percent of the total warehouse operating cost [161]. Efficient warehousing provides an important economic benefit to the business as well as to the customers. Due to the introduction of operating programs (such as cycle time reduction and quick response to orders) and new marketing strategies (e.g., micro marketing), the order picking process has become increasingly significant to manage. Moreover, catalyzed by the rapid technological advancements, the world of e-commerce is transforming fast. This significant growth in digital marketing together with the daily increase in the number of customers that buy online imposes challenges for warehouses to remain responsive as well as efficient. Any under-performance in order picking can result in high operational costs and an unsatisfactory service for the warehouse as well as the supply chain as a whole.

The order picking process is defined as *the retrieval of products from their storage locations based on customer orders*. Various activities comprise the order-picking process, including, e.g., traveling between items and packaging the order. Tompkins et al. [161] state that travelling can consume up to 50% of a picker's time, which in manual-picking systems constitutes high (labour) costs. As a result, efficient routing algorithms are needed to optimize the pick tours for retrieving the products from storage.

The problem of sequencing and finding an optimal route for a picker, i.e., obtaining the shortest tour that starts and ends at the depot and visits all items included in an order list (each item is visited exactly once) resembles the Traveling Salesperson Problem (TSP). The TSP is an NP-hard optimization problem (see Garey, Graham, and Johnson [59], Liers, Martin, and Pape [98], and Arora [9]). Within the academic literature, therefore, exact algorithms for solving TSP are only available for small instances.

In this chapter, we present an exact graph reduction algorithm for solving Steiner TSP in a planar grid graph with an application in order picker routing problem in general warehouse lay-outs using the TSP model. The TSP for order picking has a special structure in two ways. First, the warehouse lay-out provides an underlying planar graph. Second, all pick locations are found in aisles where they have degree 2 in the planar graph. Both properties are extensively used within our approach to reduce the size of the graph of the TSP. Thus, in the first step, we transform the Steiner TSP to a TSP problem by transforming the layout planar grid graph into a complete TSP graph (this graph may not be planar graph) and then, we remove a substantial amount of nodes and edges in the TSP graph. Finally, the Miller-Tucker-Zemlin formulation of the TSP is used to solve the problem to optimality.

The remainder of this paper is structured as follows. In section 2.2, we review the related studies in the literature. In Section 2.3 we provide a formal problem description and briefly represent the warehouse layout with a mathematical explanation. The graph reduction methods are provided in section 2.4. The mathematical formulation of the problem is provided in section 2.5. Furthermore, in section 2.6, the computational results are presented. The conclusions of our work are presented in Section 2.7 along with some further research suggestions.

2.2 Related literature

The order picking problem has been the subject of significant research in the past few decades, with numerous solution approaches proposed, including Dynamic Programming, Integer Linear Programming (ILP), and various heuristics. Most research has focused on modeling the problem as either a Traveling Salesman Problem (TSP) or a Steiner TSP which reduces the graph complexity and forms a more flexible network topology by the creation of additional nodes [157]. Cases where the warehouse has one or two blocks have been shown to be solvable in polynomial time.

Ratliff and Rosenthal [134] and Cornuéjols, Fonlupt, and Naddef [39] were the first to propose a dynamic programming approach for a warehouse with one block, which was polynomial in the number of items and aisles. This approach was then extended by Roodbergen and De Koster [136] to the case of two blocks and later by Cambazard and Catusse [25] to warehouses with h cross-aisles (maximum h to be solved exactly is 8); the latter being solvable non-polynomially but exponential in h.

In some papers, the Steiner TSP is used for solving this problem since it is not necessary to visit all vertices in the warehouse graph. The Steiner TSP was first studied by Cornuéjols, Fonlupt, and Naddef [39] and Fleischmann [57]. Several other formulations for the compact Steiner TSP were proposed by Letchford, Nasiri, and Theis [95], Pansart, Catusse, and Cambazard [125], Scholz et al. [147], and Valle, Beasley, and Cunha [167]. In the literature, there are three main approaches used in order to model the order picker routing problem with Steiner TSP: the first approach is transforming the Steiner TSP to TSP and then implementing MILP to reduce the size of the graph in addition to solving it. one example can be [147]. In the second approach, first, the Steiner TSP is transformed to TSP and then some graph reduction algorithm is implemented and finally the reduced graph is solved by a MILP. In the current study, this approach is implemented. In the third approach, the Steiner TSP is reduced and directly solved by an MILP (without transforming it to TSP). An example of this approach is [125]

The order picker routing problem has also been studied by modelling it as the capacitated vehicle routing problem (CVRP) in multiple studies, including Glock and Grosse [61] and Scholz and Wäscher [148]. In the CVRP formulation for this problem, each pick location is considered a node that should be visited only once by a vehicle (i.e., picker). The vehicle capacity is given by the picking trolley or forklift capacity. The objective is to find routes with the minimal total travel distance or time.

Given the limitations of the existing exact methods, in practice , the order picking problem is typically solved using heuristics. Common methods are the largest gap, return, midpoint, composite routing strategy, the combined routing strategy and finally, the S-shape method in which order pickers move in a S-shape curve along with the pick locations [62, 53, 68, 126]. A preliminary research on heuristic routing in warehouses with multiple parallel aisles was done by Hall [68]. Some of the important and commonly used heuristic algorithms are illustrated in figure 2.1.

The *S-shape strategy* leads to a route in which each aisle containing a pick location to be visited is completely traversed, and aisles where nothing has to be picked are skipped. The picker enters an aisle from one end and leaves

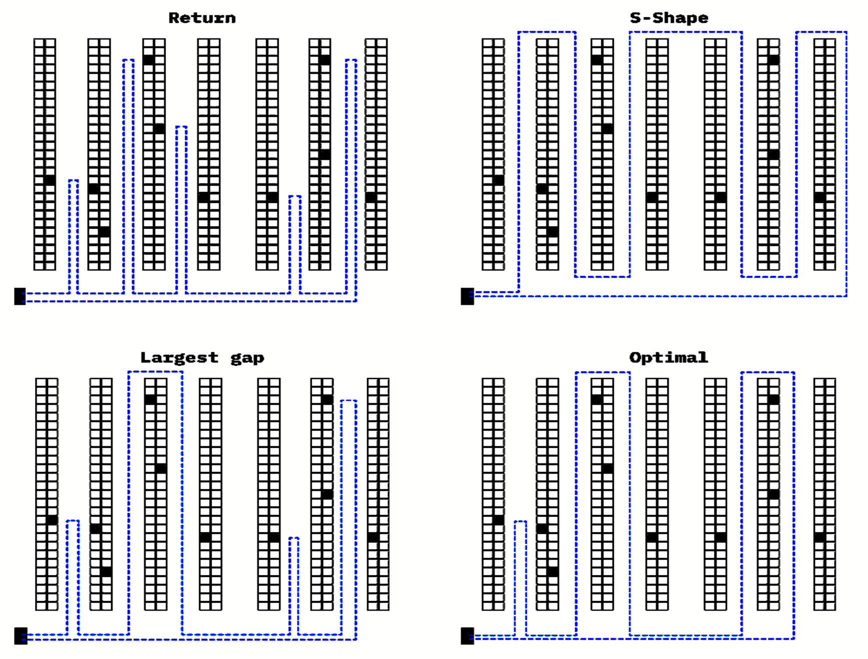


Figure 2.1: *Picker routing heuristics considered to retrieve items of a pick list (from Croucamp and Grobler [44]).*

from the other end, starting at the left side of the warehouse. After picking the last item, the order picker returns to the front end of the aisle. This S-shape strategy is used frequently, because it is very simple to use and to understand.

The *Largest Gap strategy* has the picker entering an aisle as far as the largest gap within the aisle, with a gap representing the distance between any two adjacent picks, between the first pick and the front aisle, or between the last pick and the back aisle. The largest gap is the part of the aisle that the order picker does not visit, and if the largest gap is between two adjacent picks, the picker performs a return route from both ends of the aisle. If a return route is needed, it can be taken from either the front or back aisle. This method is particularly useful when switching aisles takes little time and there are not many picks per aisle. In their study, Ho and Tseng [75] present a new way to solve the picker routing problem by combining the largest gap heuristic with a simulated annealing heuristic. Their proposed method is more efficient than the largest gap heuristic alone.

In the Return strategy, the picker enters every aisle as far as the last pick location and then returns back to the cross aisle and enters the next aisle. After picking the last item from the last aisle, the order picker returns to the front end of the aisle.

The *combined heuristic* uses both the Largest Gap and S-Shape heuristics. This means an aisle is either fully traversed or entered and exited from the same side. The best option is chosen between these two methods, and then the next aisle is entered. This process is repeated until the last item is picked, and the picker returns to the depot. The combined routing heuristic is one of the best heuristic methods available and is provided in Roodbergen [137] and Roodbergen and De Koster [136].

The S-shape and largest gap heuristics are the most commonly used routing policies in real warehouses. This is because order pickers prefer straightforward and easy-to-understand routing schemes.

In Petersen [127], more advanced heuristics are presented and their performance is compared to the optimal algorithm. De Koster and Van Der Poort [47] developed an algorithm for finding the shortest order picking routes in a warehouse with decentralized depositing [1]. In the same year, Roodbergen and De Koster [138] provide three heuristics for different situations, including a narrow-aisle warehouse used by order picking trucks. Vaughan [170] present a routing heuristic that makes use of dynamic programming for warehouses with more than two cross aisles and studied the effect of warehouse cross aisles on order picking efficiency.

Petersen and Aase [128] evaluate several picking, routing, and storage policies to determine which policy or combination of policies would provide the biggest tour reduction in total, considering four factors: picking policy, routing policy, storage policy, and average order size.

In a recent study by Weidinger, Boysen, and Schneider [176] on the picker-routing problem for mixed shelves warehouses, a nearest neighborhood heuristic method is proposed. It considers a cart pushed by the picker that allows for the assembly of multiple orders concurrently and multiple access points

[1]This refers to a layout where multiple depots and staging areas are distributed across multiple locations or areas within the warehouse instead of having a central depot.
In this setup, order pickers can deposit the collected order list directly at the nearest available depositing point rather than having to transport them to a central location.

to the central conveyor system where completed orders are handed over. Furthermore, Theys et al. [160] propose and compare the LKH (Lin–Kernighan) TSP heuristic with some of the existing heuristics in the literature such as S-shaped and largest gap and concluded that the LKH heuristic provides better solution quality (closer to optimum), although its computation time is higher.

Various metaheuristic methods have been proposed in addition to the heuristic methods discussed in the literature, such as genetic algorithms [11, 164], Ant Colony Optimization [96, 34, 48], particle swarm optimization [63, 149, 99], and tabu search [40]. Chabot et al. [33] use an adaptive large neighborhood search (ALNS) to solve the order picker routing problem and compare their proposed heuristic solution with four other existing heuristics in the literature, namely S-shape, the largest gap, the mid-point, and the combined heuristics, showing that the ALNS outperforms the other four heuristics. Bódis and Botzheim [22] applied a bacterial memetic algorithm based on pick list characteristics and order picking system characteristics to solve the order picker routing problem. Recently, Zhou et al. [182] developed three routing metaheuristics, namely a genetic algorithm, an ant colony optimization, and a cuckoo search algorithm, to solve the order picker routing problem in non-conventional fishbone warehouses with narrow aisles and a single storage system. The fishbone layout combines the conventional vertical picking rows of a warehouse with a second set of horizontal picking rows, which are separated by a V-shaped diagonal cross-aisle which traverses the entire warehouse. Ardjmand, Bajgiran, and Youssef [7] investigated the order picker routing problem using two genetic algorithms with a list-based simulated annealing. Metaheuristic methods improve the performance of the calculation method and reduce the running time by finding an approximate solution for the problem.

2.3 Problem description

In this section, we model the warehouse lay-out as a graph G_L and formulate the problem of picking orders as a shortest (closed) walk problem on G_L. Then, we define a smaller graph G_{PL} solely on the pick locations as vertices. On this graph the problem can be defined as a TSP. The TSP problem is easier to formulate and solve compared to the shortest walk problem. Afterwards, in Section 2.4, we develop ideas to reduce the number of edges and vertices in G_{PL} drastically to obtain good solution times.

2.3.1 Graph representation of the warehouse layout

Standard, multi-parallel-aisle warehouses consist of a number of longitudinal pick aisles, where product items can be picked, and intersecting cross aisles that connect these pick aisles. In practice, the cross aisles do not contain any items to pick but just allow the order picker to move efficiently from one pick aisle to another. The items are stored on both sides of the pick aisles. Order pickers are assumed to be able to traverse the aisles in both directions and to change direction within the aisles.

Each order consists of a number of items that are usually spread over multiple aisles. We assume that the items of an order can be picked in a single round. The task of a picker is to find a route (a closed walk) that starts at a depot, and then visits all picking locations, and ends at the depot again. This route should, of course, be as short as possible.

Each standard warehouse lay-out is divided in a number of blocks. A block is a row of pick aisles between two cross aisles. A detailed picture of the standard warehouse lay-out is given in figure 2.2.

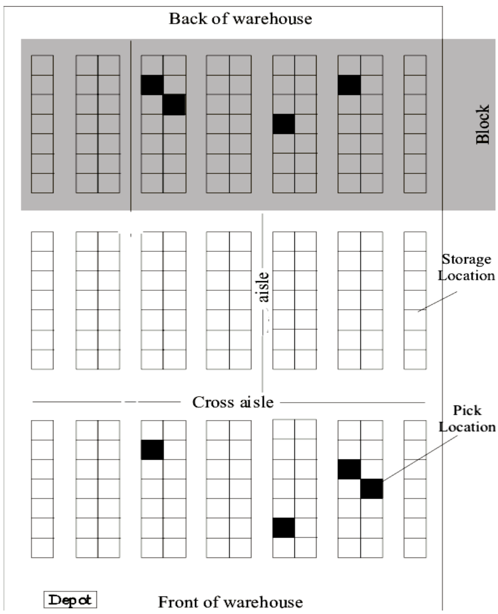


Figure 2.2: *A standard warehouse lay-out (from Roodbergen [135]).*

Though most warehouse lay-outs in the literature are indeed modelled as the standard lay-out above, this is not a limitation. Other lay-outs, such as the fish-bone lay-out, flying V, Chevron etc. have been researched too (see, e.g., Çelik and Süral [32] and Figure 2.3). For our research, the lay-out does not really matter: As long as it can be drawn in the plane (planar graph with corner points of aisles as the nodes and the aisles as the edges), our results apply. A planar graph is a graph which can be drawn on a 2D plane without any of its edges crossing each other. Please note that the requirement for the planar graph is only for the layout graph (G_L). The G_{PL} and the graph resulted from the reduction may not be planar.

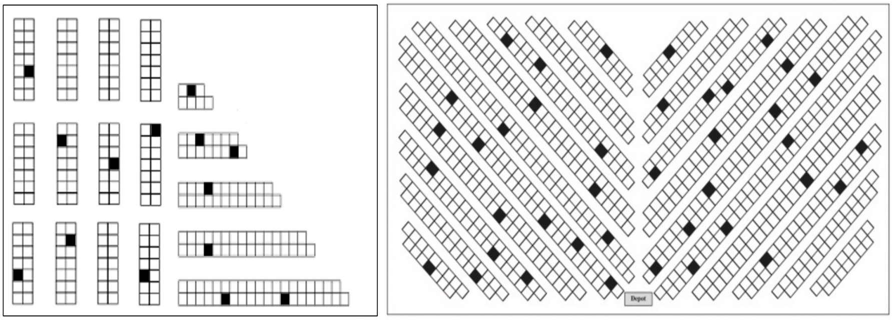


Figure 2.3: *Two other warehouse lay-outs (from Celik and Sural [30]).*

2.3.2 The pick location problem as a shortest walk problem

We model the lay-out of the warehouse with a graph $G_L = (V_L, E_L)$. Every point in the lay-out where two or more aisles meet is a vertex in V_L. Each aisle, connecting two such vertices, is represented by an edge in E_L. When picking locations are added as vertices, each edge in E_L is replaced by a path, which might be the combination of edges if the aisle contains picking locations (see figure 2.4). An edge in E_L may or may not contain picking locations. This brings us to the second part of the definition of G_L: a second set of vertices is defined by the pick locations.

For the standard lay-out described above, this results in a grid graph with symmetric distance. For the fish-bone and other warehouse lay-outs the graph is somewhat different. However, since all lay-outs are 2-dimensional physical structures, G_L is planar.

Now, every edge $\{v, w\} \in E_L$ containing pick locations is replaced by a path as follows. Suppose that the edge $\{v, w\}$ contains $j-1$ pick locations $(p_1, \ldots, p_{j-1})$. Let $v = p_0$ and $w = p_j$. We now replace the edge $\{v, w\}$ with the path $v = p_0, p_1, \ldots, p_j = w$. So, besides adding the vertices $p_1, \ldots, p_{j-1}$, we also add the edges $\{p_{i-1}, p_i\}$, $(i = 1, \ldots, j)$ to E_L. Note that the vertices representing pick locations have degree 2 — which is important for the remainder of our analysis.

A special vertex in V_L is the depot, where we start and end the pick tour.

The graph G_L for two example lay-outs is illustrated in Figures 2.4 and 2.5. Here, black dots represent pick locations and white dots represent corner points of aisle.

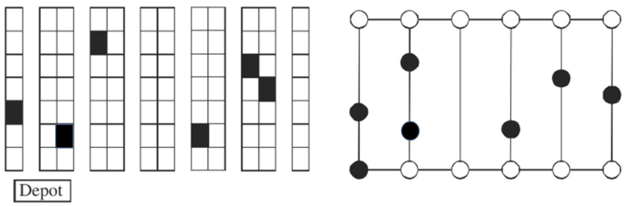


Figure 2.4: *Graph G_L for standard warehouse lay-out (from [135, 32]).*

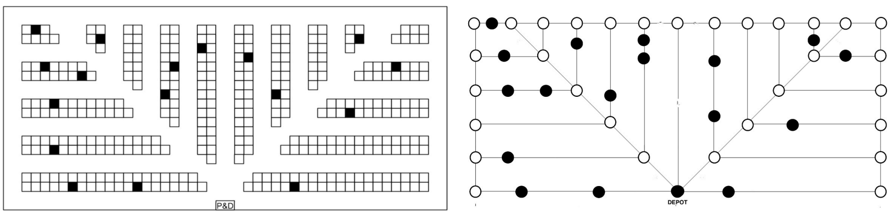


Figure 2.5: *Graph G_L for a fish-bone warehouse lay-out (from [32]).*

Each edge $\{v, w\} \in E_L$ has a length d_{vw}, representing the actual distance between the vertices v and w in the warehouse lay-out. Thus, this is simply the length of the edge in the lay-out connecting v and w.

We then define the order picking problem as follows: find a shortest walk in G_L that starts and ends at the depot, visiting each vertex, representing a pick location in V_L, at least once.

2.3.3 Modelling the pick location problem as a TSP

The problem can be modelled as a TSP on a graph, containing only the nodes of the pick locations. This graph $G_{PL} = (V_{PL}, E_{PL})$ only contains the vertices of G_L that represent pick locations and the depot. Each edge $\{v, w\} \in E_{PL}$ has a length that is the shortest distance between the pick locations v and w in the lay-out graph G_L. Please note that G_{PL} may no longer stay planar. In G_{PL}, the problem is to find the shortest tour through all vertices, i.e., the standard TSP problem. An example of transforming G_L to G_{PL} is illustrated in figure 2.6

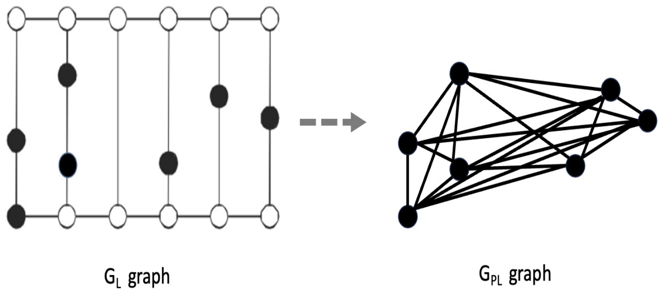


Figure 2.6: *Transforming G_L to G_{PL}.*

Distances in G_{PL} if the underlying graph G_L is from a standard lay-out

Consider the standard lay-out (see again Figure 2.4), where G_L is a grid graph. Let (x_i, y_i) be the coordinates of product location i, $(i = 1, \ldots, |V_L|)$. Without loss of generality and to simplify notation we assume the direction of x and y align with the cross aisle and pick aisle, respectively. To travel between two product locations i and j, the picker will always prefer the shortest path. To calculate the length of each edge in G_{PL}, which is the shortest path between the two vertices (either a pick location or the depot) of the edge, we do the following:

1. If both nodes (product locations) are in different blocks (different row of aisles) the shortest distance between the two nodes is calculated as the Manhattan distance, which is the rectilinear route measured along

parallels to the horizontal and vertical axes of the plane. The Manhattan distance between two points with coordinates (x_1, y_1) and (x_2, y_2) is

$$d_{12} = | x_1 - x_2 | + | y_1 - y_2 | .$$

2. If both nodes (product locations) are in the same block (same row of aisles), the shortest path has to go through one of the two cross aisles adjacent to the block. The length of both possible paths is then determined and the shorter one is kept. Therefore, for two points with coordinates (x_1, y_1) and (x_2, y_2) we will have

$$d_{12} = \{min\{\gamma_1 + \gamma_2, \beta_1 + \beta_2\} + | x_1 - x_2 |\} = \{| x_1 - x_2 | + | y_1 - y_2 | + 2 \times min\{\gamma_1, \gamma_2, \beta_1, \beta_2\}\}.$$

Here, γ_i, $(i = 1, 2)$, is defined as the difference between the y_i-coordinate of each location with the cross node located above it, and β_i, $(i = 1, 2)$, is defined as the difference between the y_i-coordinate of each location with the cross node located below it. A graphical sketch of the warehouse lay-out for this case is illustrated in figure 2.7.

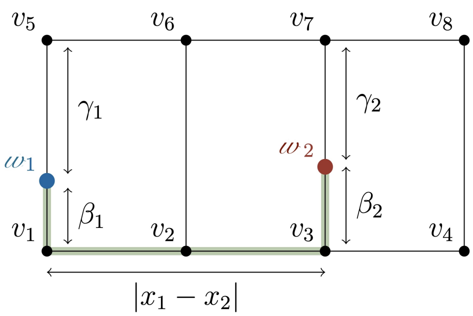


Figure 2.7: *Distance of vertices in the same block, but different aisles.*

2.4 Properties of the lay-out graph G_L and the pick locations graph G_{PL}

In this section we do two things: reduction of vertices and reduction of edges in G_{PL}. First, we show that some pick locations need not be present in G_{PL}.

Second, we show that there is an optimal walk in G_L that can be translated into an optimal tour in G_{PL}, using only a very small portion of the edges in G_{PL}. Roughly speaking, we only need the edges of G_{PL} that correspond to shortest paths in G_L with at most one intermediate pick location.

2.4.1 Visiting pick locations in G_L

Following Pansart, Catusse, and Cambazard [125], there are three different ways by which items in each aisle in a warehouse can be picked, as illustrated in figure 2.8:

1. Visit the complete aisle in one direction (two ways). See 2.8a, and 2.8b.
2. Enter the aisle from one of the corner nodes until you reach the last product location on the aisle and then return to the same corner node (two ways). See 2.8c, and 2.8d.
3. Let the biggest gap between two pick locations be the one between p_{i-1} and p_i. Enter the aisle from corner node p_0 to p_{i-1} and return, and from corner node p_j to p_i and return. See 2.8e.

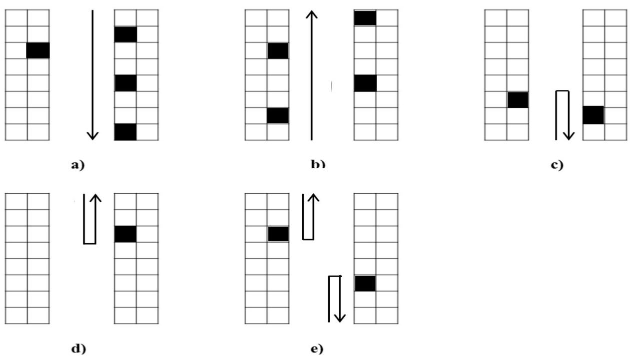


Figure 2.8: Five different ways for aisle traversal (from [125]).

Consequently, there are at most 4 pick locations in each aisle [125]: the ones closest to the corner nodes and the ones that have the biggest gap between them. Note that there can be fewer than four relevant pick locations on an aisle, or some of the nodes could coincide. When removing irrelevant nodes one should pay attention to the situation where the biggest gap may move. (See figure 2.9) Meaning that by removing a node, the gap between the remaining nodes in the aisle may get bigger than the initial gap. In that case we take care of that in the model by adding constraints that force necessary edges to be in the walk. To do this, we use the algorithm proposed by [125] and do the following:

for every aisle **do**

- Compute the largest gap between two vertices

- Identify the two sets containing all products below and above the largest gap (call them set T and set S). These sets can be empty or singleton.

- In each subset, keep the two products that are the farthest apart ($t_1, t_2 \in T$ and $s_1, s_2 \in S$)

- Add the constraints forcing the order picker to traverse each set once. (In the mathematical model it translates into the following constraints: $(\{s_1, s_2\} + \{s_2, s_1\} >= 1$ if $s_1 \neq s_2$ and $\{t_1, t_2\} + \{t_2, t_1\} >= 1)$ if $t_1 \neq t_2$.

In these constraints, $\{s_1, s_2\}$ refers to the edges in set S connecting first and second farthest products.

end

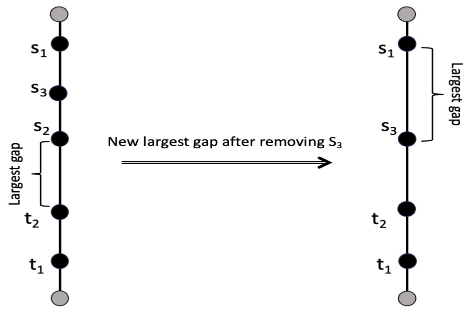


Figure 2.9: The change of largest gap and removing extra nodes in an aisle

2.4.2 Properties of an optimal walk in G_L

Lemma 2.4.1. *An optimal walk in $G_L = (V_L, E_L)$ that visits all pick locations at least once, will not use any edge in G_L twice, or more, in the same direction.*

Proof. Consider a walk W. Let the edge $e = \{v, w\} \in E_L$ be used twice (or more) in the direction $v \to w$ by W. Then, W contains two paths from w to v, say P_1 and P_2. The walk is now: $W = (v, w, P_1, v, w, P_2, v)$. Now, consider the walk $W' = (w, P_1, v, P_2^{-1}, w)$, where P_2^{-1} is the path P_2 traversed in backward direction. W' is shorter than W as it does not use the edge e anymore. Moreover, it visits every vertex that W visits, using edges from W. Concluding, W cannot be optimal. The proof is visualized in Figure 2.10. □

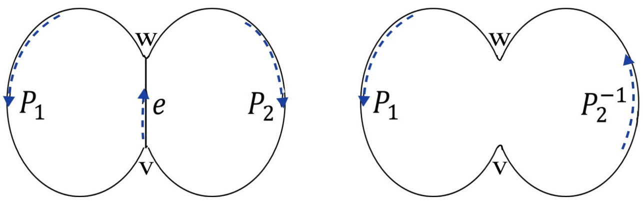


Figure 2.10: *Edge twice in same direction*

In Kai and Chuanhou [81], a somewhat more complicated proof is given of this lemma. From Lemma 2.4.1, we can conclude that in an optimal walk, every edge is traversed at most twice. And if twice, then in opposite directions.

Lemma 2.4.2. *If an optimal walk in $G_L = (V_L, E_L)$ uses the edge $e = \{v_e, w_e\}$ twice — in opposite direction, as a consequence of the previous lemma — say, $W = (v_e, P_1, v_e, w_e, P_2, w_e, v_e)$, then the paths P_1 and P_2 have no vertices of G_L in common.*

Proof. Suppose that a walk W contains $e = \{v_e, w_e\}$ twice (in opposite direction), and that a vertex $v \neq v_e, w_e$ is visited by both paths P_1 and P_2. Then the walk can be described as

$$W = (v_e, P_1^{'}, v, P_1^{''}, v_e, w_e, P_2^{'}, v, P_2^{''}, w_e, v_e)$$

Here: $P_1 = (P_1^{'}, v, P_1^{''})$ and $P_2 = (P_2^{'}, v, P_2^{''})$, where

$P_1^{'}$ is a path from v_e to v.

$P_1^{''}$ is a path from v to v_e.

$P_2^{'}$ is a path from w_e to v.

$P_2^{''}$ is a path from v to w_e.

From W we create a better walk W' as follows:

$$W' = (v_e, P_1^{'}, v, P_2^{''}, w_e, P_2^{'}, v, P_1^{''}, v_e)$$

W' visits all vertices of W, but does not use the edge $e = \{v_e, w_e\}$ anymore and thus, it is shorter. See a visual of this proof in Figure 2.11. □

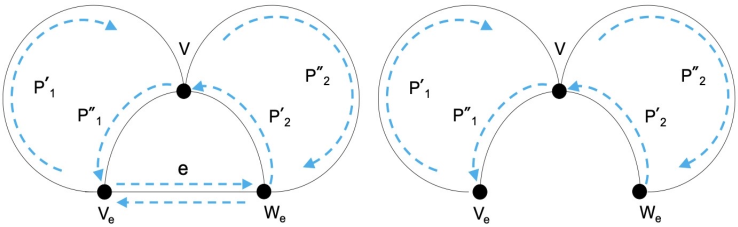


***Figure 2.11:** No alternating edges*

As a corollary we now have the following Theorem.

Theorem 2.4.3. *Consider an optimal walk W in $G_L = (V_L, E_L)$ that visits the pick location v_1 twice. If there is a vertex v in W, also visited twice, then these locations are not visited in an alternating way, i.e., $W = (v_1, P_1, v, P_2, v_1, P_3, v, P_4, v_1)$ is not possible.*

Note that P_1 and P_3 denote paths from v_1 to v and P_2 and P_4 are paths from v to v_1.

The proof of this theorem leans on the fact that the pick location vertices have degree 2 in G_L. This applies for any practical lay-out, due to the fact that the pick location vertices lie on a path representing an aisle between the two corner vertices of the aisle.

Proof. If vertex v_1 is visited twice, then both incident edges must be used twice (each in opposite direction): If only one incident edge is visited twice, or both edges are visited once, then v_1 is visited only once.

Consider one of the two edges, say $e_1 = \{v_1, v_2\}$. Now, the optimal walk W can be described as $W = (v_1, P_1, v_1, e_1, v_2, P_2, v_2, e_1^{-1}, v_1)$. According to lemma 2.4.2, the paths P_1 and P_2 have no vertex in common. Thus, a vertex $v \neq v_1, v_2$ that occurs twice in W, does so either twice in P_1 or twice in P_2. □

This brings us to the conclusion that all vertices in the optimal walk that are visited twice are not alternating. For a specific vertex, say v, this means that, in the walk (v, P_1, v, P_2, v) the occurrence of the other vertices visited twice are either both in P_1 or both in P_2. Thus, the occurrence of the vertices has a nested structure. See the visualization in Figure 2.12. In this figure, the circle represents the complete walk W. Each piece between two connected vertices (red lines) on the circle represents a subpath of W, with only vertices that are visited exactly once by W.

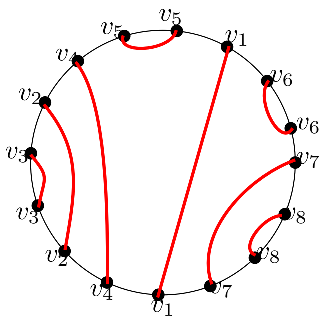


Figure 2.12: *No alternating vertices (the nested structure of the occurrence of the vertices in the walk*

In case a vertex v has degree higher than 2, theorem 2.4.3 does not apply, since the theorem uses that if a node is visited twice then an edge is visited twice, and this need not be true when a vertex has degree higher than 2.

However, for a node with degree higher than two, we can take one of its incident edges and move it up that edge a small amount ϵ. Then, we can apply the previous theorem again (see figure 2.13), as all vertices of G_{PL} have degree 2 again in G_L. This may not be useful to the picker routing problem that we treat, but it is helpful for the TSP problem in general planar graphs.

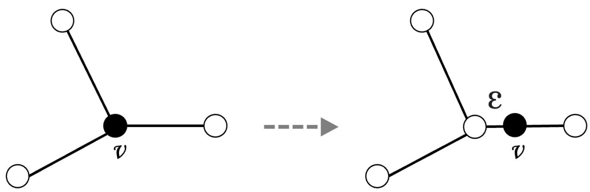


Figure 2.13: *From degree > 2 to degree = 2*

2.4.3 From an optimal walk in G_L to an optimal TSP tour in G_{PL}

Figure 2.12 illustrates an example of how pick locations appearing twice could be on the optimal walk W in G_L. Each connection between two such vertices is a path in G_L with possibly some vertices that occur exactly once. We shall transform the walk W into a tour on the pick locations only, e.g., the vertices in G_{PL}. In doing so we only use a limited set of edges in G_{PL}.

We start with removing the vertices of W that are not pick locations. Every sub-path that begins and ends with a pick location — with no other intermediate pick locations — is replaced by a single edge. This results in a walk W' in G_{PL}: the vertices are only pick locations, and the edges connect two pick locations. This walk is optimal as the "contracted" sub-paths are shortest paths in G_L with the same length. The pick locations that occur twice in W also appear twice in the new walk W'. Our next step is to remove one of the two occurrences of these vertices.

First, we number the occurrences of these doubly occurring vertices in the order that we visit them on the walk W'. See figure 2.14. Now note that due to theorem 2.4.3, the two occurrences have different parity: one has an odd number, the other an even number. Next, each occurrence of a vertex with an even number is removed by connecting the neighbor pick locations (independent of whether these occur once or twice in the walk) with a direct edge. Note that this edge is present in G_{PL}. Now, the new walk contains only edges of E_{PL}. Moreover, the edges that we use have at most one pick location on the path in G_L. Finally, each pick location is now visited exactly once. Thus, not only is the walk a tour in G_{PL}, but it only uses edges on which at most one intermediate pick location is present. Thus, in solving the TSP problem of G_{PL}, we can restrict ourselves to using such edges only.

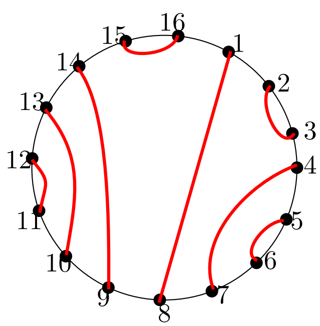
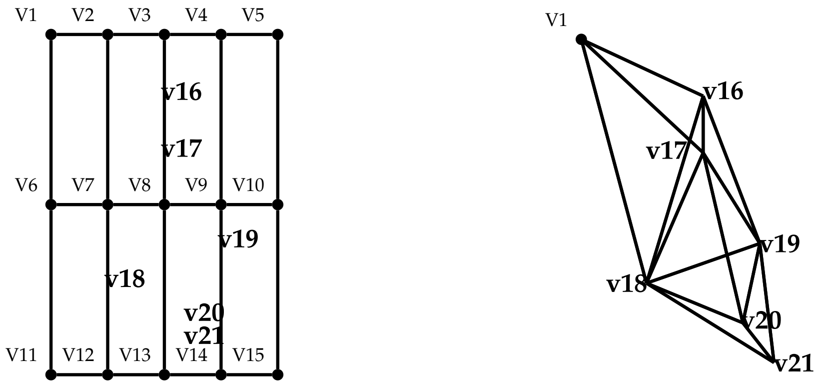


Figure 2.14: *No alternating vertices.*

In figure 2.15, we translated the lay-out graph G_L to the graph G_{PL} with only the edges needed to find an optimal TSP tour. The figure illustrates how the graph G_{PL} is reduced to a graph with a much smaller amount of edges than in the complete graph G_{PL}. The depot node V1, for example, is not connected to the product locations $v_1 9$, $v_2 0$ and $v_2 1$, since there exist more than one pick locations on the possible shortest paths between them.



***(a)** G_L warehouse layout.*

***(b)** G_{PL} with reduced set of edges.*

***Figure 2.15:** Graph reduction example*

2.5 Mathematical formulation

We now formulate the proposed exact picker routing model described above using the model presented in [114]. In this model, we are looking for the shortest tour in G_{PL} such that the order picker visits all the ordered product locations only once, starting and ending from the depot. V_{PL} refers to all the vertices of the G_{PL} (product locations). The problem is formulated as a traveling salesman problem (TSP). *Remark:* Common practice is that nodes in mathematical formulations are described with indices i and j, rather than the descriptions v and w. In the sequel we shall do that as well. Nodes 0 and $n + 1$ are considered to be the visiting order of the depot in the beginning and the end of the tour. The mathematical model for the general TSP problem is as follows:

Table 2.1: Notation for TSP model.

Parameters	
d_{ij}	Distance between nodes indexed i and j.
Decision variables	
x_{ij}	Binary variable, 1 if arc (i, j) is traversed by the picker; 0 otherwise.
u_i	Position of node i in the pick tour.

$$\min \sum_{i,j} d_{ij} x_{ij} \tag{2.1}$$

$$s.t.$$

$$\sum_{i:i \neq j} x_{ij} = 1 \qquad \forall j \in V_{PL} \tag{2.2}$$

$$\sum_{j:j \neq i} x_{ij} = 1 \qquad \forall i \in V_{PL} \tag{2.3}$$

$$u_i - u_j + n * x_{ij} \leq n - 1 \qquad \forall i, j \in V_{PL} \tag{2.4}$$

$$u_0 = 0 \tag{2.5}$$

$$u_{n+1} = n + 1 \tag{2.6}$$

$$u \in \mathbb{Z}, x_{ij} \in \{0, 1\} \qquad \forall i, j \in V_{PL} \tag{2.7}$$

The objective function (2.1) minimizes the total distance travelled by the picker. Constraints (2.2) and (2.3) force the order pickers to enter and exit every location exactly once, respectively. Constraints (2.4) eliminates sub-tours by allocating an index to each visited location. In constraints (2.5) and (2.6) the depot is initiated as the start and the end of the tour, respectively. Finally, constraints (2.7) set the domain of the decision variables.

2.6 Model implementation and numerical results

To test the performance of our graph reduction approach, we perform a set of computation experiments on two different benchmark, single block instances from Scholz et al. [147] and multi-block instances from Theys et al. [160].

The mathematical model presented in this paper is implemented in GAMS 24.2 and solved using IBM CPLEX 12.6. All experiments are run on on a 11th Gen Intel(R)Core i5-1135G7@2.0GHz and 8GB of RAM. The computing time for each instance is limited to 30 minutes. The CPLEX version and the computer used is well-matched with the ones used in [147] and [125], so that the computational time comparison is fair.

2.6.1 Single block: comparison with instances presented in [147]

The numerical results using the first benchmark instances from [147] containing only a single block (two cross aisles) are given in table 2.3. In this table, the two parameters given as m and n, are the number of aisles and the number of items in the order list respectively. Furthermore, the column $R - edges$ refers to the number of edges in the graph after the reduction is implemented. We solved each instance 10 times and put the average results in the table 2.3. The numerical results of from [147] are presented in the last column.

As shown in this table, the average graph reduction in comparison to the TSP model (complete graph without reduction) is 72.85%, which is significant. Furthermore, for all of the instances, our model gives the optimal solution almost instantaneously. The maximum computation time for the biggest instance (with 30 aisles and 90 products) is 7.6 seconds, which is considerably less compared to 786.29 seconds by [147].

Furthermore, the authors extended their work to present a new MILP and graph reduction method for the larger warehouses with 2 and 3 blocks. [146]. However, for some of the larger instances with more than 15 aisles, their algorithm is not able to find the optimal solutions within the time limit of 30 minutes. More precisely, for 10% of the instances with only 3 blocks, no optimal solutions were reported. However, we will show in the next section that our model with the help of the pre-processing phase is able to solve much larger instances in a very short time.

2.6.2 Multi-block: Comparison with the instances presented in [125]

The second benchmark is from [125] and consists of 9 different scenarios: three different numbers of aisles (5,15,60), three different numbers of cross-aisles (3,6,11) and three different numbers of products in the order (15,60,240). We solve these instances on the multi-block warehouse. The results are presented in table 2.4. In this table, the column ”Concorde” refers to the running time using Concorde solver on a complete graph (before reduction) and the column ” Concorde+” refers to the running time using Concorde on the reduced graph. This helps us better understand the effect of our reduction algorithm on the Concorde solver. It is shown that our algorithm is still

Table 2.3: Numerical results of proposed model on different instances

m-n	Time (s)	VarNum	ConNum	Initial edges	R-edges	Reduction(%)	Shcolz(s)
5-30	0.031	68	93	435	21	95	0.09
5-45	0.046	76	101	990	29	97	0.09
5-60	0.063	84	109	1770	41	97	0.09
5-75	0.081	92	116	3492	50	98	0.09
5-90	0.109	100	125	4005	64	98	0.10
10-30	0.047	256	301	435	136	68	1.60
10-45	0.110	274	319	990	145	85	1.03
10-60	0.125	292	337	1770	191	89	1.42
10-75	0.281	360	413	2775	213	92	1.36
10-90	0.906	376	423	4005	220	94	0.62
15-30	0.479	456	504	435	167	61	2.29
15-45	1.187	576	631	990	247	75	5.28
15-60	2.297	620	685	1770	346	80	10.64
15-75	2.337	698	782	1914	391	79	15.10
15-90	2.508	730	813	2106	438	79	19.41
20-30	1.225	438	484	435	315	27	10.57
20-45	1.391	812	878	990	349	64	27.32
20-60	1.651	1016	1091	1770	476	73	114.33
20-75	1.981	1030	1108	2775	495	82	216.63
20-90	3.023	1046	1129	4005	679	83	485.71
25-30	0.911	618	672	435	206	52	54.46
25-45	2.160	1090	1178	990	377	62	85.46
25-60	2.425	1270	1353	1770	588	66	258.92
25-75	3.063	1346	1437	2775	703	74	527.39
25-90	4.469	2106	2213	4005	939	76	646.59
30-30	3.234	626	681	435	348	20	204.18
30-45	5.547	1176	1250	990	511	48	406.19
30-60	6.328	1906	2007	1770	627	64	508.80
30-75	7.042	2409	2521	2775	833	70	638.89
30-90	7.601	2862	2981	4005	997	75	786.29

very effective when using Concorde solver by reducing the running times. The column "Total arcs" and "Reduced arcs" refer to the number of arcs in our graph, before and after implementing the proposed reduction algorithm. As it is shown in the table 2.4, our method is able to solve all the multi-block instances optimally in less than 70 seconds with CPLEX and in less than 5 seconds with Concorde.

Table 2.5 illustrates the number of unsolved instances within the 30 minutes time limit both for [125] and our proposed method. In this table, SCFS+ stands for the standard single commodity flow formulation with pre-processing and the additional valid inequalities and PDYN refers to a dynamic program, both proposed by [125].

As it is shown in this table, [125] are not able to solve larger instances with 240 products or more than 6 cross-aisles (15-11-240, 5-11-240, 60-11-240, 60-3-240, 60-6-240). Furthermore, their proposed dynamic program is very efficient for the small instances, however, it is not scalable and it becomes intractable for instances with more than 6 cross-aisles. Our model, however, is able to solve all the instances to optimality within the time limit.

Moreover, the average computation time (in seconds) for these instances are presented in table 2.6. In this table, SCFS+, PDYN, CDE and CDE+ refer to the ingle commodity flow formulation, dynamic programming, Concorde and Concorde after reduction times respectively in [125]. CPLEX+, our CDE and Our CDE+ refer to the running times with CPLEX using reduction method, Concorde times before and after implementing our reduction algorithm respectively. Our Concorde times before reduction is lower that the ones in [125]. However, it can be seen that the Concorde times after reduction for both our reduction and algorithm and the one presented in [125] are equal on average. It is shown that the average running time for the largest instance solved in less than 30 minutes for [125] is 111.64 seconds which is 3 times larger than our results in CPLEX. Comparing the running times using CPLEX solver, the average computation time for our model is 16.40 seconds for all the instances which is 55% lower than their proposed single commodity flow formulation solved by CPLEX.

It is worth mentioning that the lower total average time for the dynamic program model for [125] is due to the fact that this model is only able to solve the smaller instances. Moreover, dynamic programming is not ideal for problems with constraints such as precedence, flow directions, or multiple depots: a MILP better addresses such requirements [125]. Furthermore, horizontal and vertical components employed in dynamic programming cannot be used in general warehouses (does not consider aisles), however our algorithm is applicable on any warehouse layout with planar graph. Furthermore, the SCFS+ model proposed in [125] uses "warm start" or an initial upper-bound obtained by LKH (Lin–Kernighan–Helsgaun) algorithm which affects their computational time. There are other aspects in addition to the running time, which make our algorithm more effective. The main advantage of our method

Table 2.4: Numerical results of the proposed model on different instances

Aisles	Cross aisles	Products	CPLEX(sec)	Concorde(sec)	Concorde+(sec)	Total arcs	reduced arcs	reduction%
5	3	15	0.25	0.02	0.01	105	61	41.9
5	3	60	1.84	0.22	0.18	1770	375	78.8
5	3	240	5.36	1.57	1.09	28680	797	97.2
5	6	15	0.14	0.02	0.01	105	67	36.2
5	6	60	2.16	0.17	0.09	1770	399	77.5
5	6	240	11.3	3.81	1.28	28680	1188	95.9
5	11	15	0.23	0.03	0.02	105	72	31.4
5	11	60	5.97	0.16	0.09	1770	403	77.2
5	11	240	15.42	5.87	3.95	28680	1309	95.4
15	3	15	0.27	0.01	0.01	105	77	26.7
15	3	60	9.28	0.29	0.16	1770	436	75.4
15	3	240	25.32	7.23	4.72	28680	4062	85.8
15	6	15	0.09	0.03	0.02	105	82	21.9
15	6	60	10.27	0.31	0.26	1770	567	68.0
15	6	240	42.09	8.21	5.23	28680	4139	85.6
15	11	15	0.38	0.02	0.01	105	83	21.0
15	11	60	19.38	0.22	0.16	1770	556	68.6
15	11	240	49.32	9.52	3.77	28680	4454	84.5
60	3	15	0.28	0.03	0.02	105	81	22.9
60	3	60	14.28	0.16	0.12	1770	656	62.9
60	3	240	55.92	10.34	5.81	28680	7974	72.2
60	6	15	0.25	0.01	0.01	105	83	21.0
60	6	60	18.55	0.34	0.18	1770	734	58.5
60	6	240	61.48	11.18	5.75	28680	9468	67.0
60	11	15	0.19	0.01	0.01	105	82	21.9
60	11	60	24.33	0.29	0.23	1770	858	51.5
60	11	240	68.24	12.19	4.41	28680	10374	63.8

Table 2.5: Number of unsolved instances after 30 minutes. Benchmark against [125]

	Total	# aisles			# cross-aisles			# products		
		5	15	60	3	6	11	15	60	240
SCFS+	18	1	4	13	1	2	15	0	0	18
PDYN	90	30	30	30	0	0	90	30	30	30
Our method	15	0	6	9	4	5	6	0	1	14
# instances	270	90	90	90	90	90	90	90	90	90

is the simplicity of the model (TSP), with lower number of constraints and variables and it can be easily implemented without using complex computer programs.

We further compare the number of arcs and the reduction percentage for our proposed graph reduction algorithm and the one presented in [125]. The results are presented in table 2.7. As it is shown in this table, our proposed graph reduction algorithm performs really well with an average reduction of 60% for all the instances. When comparing the number of arcs in the graph, we can see that these numbers both for before and after reduction are very higher (on average 3.6 times higher) than the number of arcs in our graph which is very considerable. In the algorithm proposed by [125], the number of arcs after reduction are on average 33% of the total arcs before reduction. However this number for our algorithm is 15% which is much lower.

Table 2.6: *Average computation time (seconds) for instances solved in 30 minutes. Benchmark against [125].*

	Total	# aisles			# cross-aisles			# products		
		5	15	60	3	6	11	15	60	240
SCFS+	36.07	23.89	56.96	27.12	3.44	35.88	71.69	0.07	4.05	111.64
PDYN	0.27	0.05	0.16	0.61	0	0.54	-	0.24	0.27	0.30
CDE	6.86	17.61	2.11	0.88	14.82	3.89	1.88	0.01	0.13	20.45
CDE+	1.6	3.45	1.04	0.30	2.20	1.55	1.04	0.01	0.1	4.68
Our Cplex+	16.40	4.85	17.37	27.05	12.53	16.27	20.38	0.23	11.78	37.16
Our CDE	4.34	4.21	3.98	4.84	3.87	5.34	3.81	0.02	0.24	12.77
Our CDE+	1.6	1.08	1.93	2.25	1.78	1.98	1.52	0.01	0.16	4.00

Table 2.7: *Average number of arcs in TSP graph and Steiner graph, with and without pre-processing. Benchmark against [125].*

	Total	# aisles			# cross-aisles			# products		
		5	15	60	3	6	11	15	60	240
TSP	20580	20580	20580	20580	20580	20580	20580	240	3660	57840
TSP+	6676	2319	6042	11665	3744	7238	9046	217	2350	17460
Reduction%	68	89	71	43	82	65	56	9	36	70
Our TSP	11018	11435	11435	11435	11435	11435	11435	105	1770	28680
Our TSP+	1830	519	1606	3367	1613	1858	2021	76	553	4862
Our Reduction%	60	70	60	47	63	59	57	27	69	83

Table 2.8: *Numerical results of the proposed model on different instances with three blocks and comparison with [146]*

m-n	CPLEX Time(sec)	Scholz time(sec)
5-30	0.51	0.37
5-45	0.63	0.40
5-60	0.76	0.44
5-75	0.90	0.47
5-90	0.106	0.56
10-30	0.71	1.57
10-45	0.88	1.83
10-60	0.94	1.79
10-75	1.29	2.31
10-90	2.37	3.16
15-30	4.89	14.00
15-45	6.27	6.44
15-60	7.69	7.20
15-75	10.73	17.79
15-90	12.04	14.00
20-30	6.02	7.25
20-45	8.45	17.06
20-60	11.22	66.37
20-75	13.09	100.81
20-90	14.64	108.43
25-30	7.18	18.47
25-45	9.27	22.50
25-60	13.64	93.28
25-75	15.07	170.96
25-90	16.72	205.47
30-30	11.08	71.38
30-45	14.50	58.65
30-60	16.66	232.28
30-75	18.03	155.85
30-90	21.78	339.70

2.7 Concluding Remarks

This study aimed to develop an efficient algorithm for the planar TSP, specifically for the order pickers routing problem in a multi-block warehouse layout. While exact algorithms for small warehouses exist in the literature, solution algorithms for larger warehouse layouts often rely on (meta)heuristic approaches.

The proposed algorithm utilizes graph reduction to eliminate unnecessary vertices and edges from the graph (The network size reduction achieved by our algorithm for a single block warehouse was, on average, 72.85% and for multi-block warehouses 60%), resulting in a significant reduction in computation time. This approach is applicable to any warehouse layout presented as a planar graph, making it practical for real-world applications.

To solve the routing problem, a general TSP model was used. The algorithm was implemented on various problem instances from the literature, and its performance was compared with existing methods. The results showed that the proposed algorithm performs really well and compared to the existing methods in the literature, our method is better in terms of simplicity, size, and calculation time. Overall, the results indicate that the proposed algorithm is a promising approach for solving the Steiner TSP in a planar graph with an application in order pickers routing problem in multi-block warehouse layouts. This algorithm can be applied on any other problem with the same graph structure which makes it very efficient and practical.

3

The order picking problem under a scattered storage policy

Adapted from: [. Rajabighamchi, van Hoesel, and Defryn [133]]

Abstract

When warehouses are operated according to a scattered storage policy, each Stock Keeping Unit (SKU) is stored at multiple locations inside the warehouse. Such a configuration allows for improved picking efficiency, as now an SKU can be picked from the location that is most compatible with the other SKU's in the picking batch. Seizing these benefits, however, comes at the cost of additional decisions to be made while planning the picking operations. Next to determining the sequence in which SKU's will be retrieved from the warehouse, the location at which each SKU needs to be extracted has to be chosen by the planner. In this chapter, we model the order picking problem under a scattered storage policy as a Generalized Travelling Salesperson Problem (GTSP). In this problem, the vertices of the underlying graph are partitioned into clusters from which exactly one vertex should be visited in each cluster. In our order picking application, each cluster contains all product locations of a single SKU on the order list. The aim is to design a pick tour that visits all product locations of the SKU's on the pick list (i.e., visit each cluster exactly once) and minimizes the total travel distance. We present an ILP formulation of the problem and a variable neighbourhood heuristic, embedded in a guided local search framework. The performance of both methods is tested extensively by means of computational experiments on benchmark instances from the literature.

3.1 Introduction

In recent years, the advent of e-commerce and the relentless pursuit of customer satisfaction have reshaped the logistics and warehouse management paradigms. Warehouses are no longer perceived merely as storage spaces but as dynamic hubs requiring precision and agility in their operational strategies. One critical aspect of warehouse management that significantly influences overall efficiency is the order picking process. As the heartbeat of warehouse operations, order picking involves retrieving products (in the remainder of the chapter referred to as SKUs) from storage locations to fulfill customer orders accurately and promptly [46, 168]. The choice of a storage policy profoundly impacts the order picking process, and one such policy that has gained attention for its potential benefits is the Scattered Storage Policy, where items are dispersed across the warehouse based on factors such as demand patterns, product characteristics, and frequency of retrieval.

The problem of finding the optimal pick tour (i.e., a minimum cost / length tour that starts and ends at the depot and visits all SKU locations from the current pick list) is closely related to a variant of the Travelling Salesperson Problem (TSP). Often, a Steiner TSP formulation is used, in which aisle intersections are added as intermediary nodes that could (but should not) be part of the tour.

In this chapter, we study the order picking problem in combination with a scattered storage policy [74]. As nowadays warehouse operations are commonly supported by warehouse management software, there is no real need to group all SKUs of a certain type at the same storage location. Incoming SKUs can be put away at any available storage location from which they can be retrieved once ordered. Consequently, SKUs are *scattered around the different storage locations* and each SKU becomes available at different locations within the warehouse and each storage location contains only one SKU.

The adoption of the Scattered Storage Policy is driven by several compelling advantages. First, it introduces enhanced flexibility and adaptability, allowing warehouses to easily adjust to changes in product demand, inventory size, and SKU variability. Unlike traditional centralized storage systems, scattered storage permits a decentralized arrangement, optimizing space utilization and maximizing storage capacity. The dispersion of products minimizes the likelihood of congestion and bottlenecks, leading to smoother order picking processes and enhanced overall warehouse throughput. Moreover, scattered storage improves accessibility and retrieval speed, strategically distributing items for quicker retrieval. This not only reduces travel distances for order pickers but also contributes to faster order fulfillment times, ultimately enhancing customer satisfaction through timely deliveries. Moreover, the Scattered Storage Policy facilitates adaptation to variable demand profiles, allowing warehouses to dynamically respond to changes in customer preferences and market trends.

Under such scattered storage policy, the construction of the pick tours combines the decision on the storage location from which the required SKUs will be picked with the generation of optimal pick tours in which these selected locations are visited. As out of all available storage locations for each SKU the decision maker can select the location that is *most compatible* with that of all other SKUs to be picked within the same tour, more efficient pick tours can be constructed.

This chapter contributes to the academic literature on warehouse operations in the following ways:

- Denoting all storage locations from which an SKU can be picked as a cluster from which exactly one location should be visited to pick the corresponding SKU, we formulate the order picking problem under a scattered storage policy as a Generalized Travelling Salesperson Problem (GTSP).
- We present an improved ILP formulation for the GTSP and prove its performance against state-of-the-art formulations by means of extensive computational experiments.
- To allow for scalability and solve larger problem instances fast, we present a guided local search heuristic for which we obtain very competitive results.

The remainder of the chapter is organized as follows: we discuss the relevant literature in section 3.2. In section 3.3 we formally define the generalized traveling salesman problem for the order picking problem under a scattered storage policy. Section 3.4 details our Guided Local Search algorithm for solving the problem after which we test the performance of our formulations and benchmark our heuristic solution approach against the current state-of-the-art in Section 3.5. Finally, we summarize our main conclusions in Section 3.6.

3.2 Literature Review

3.2.1 The order picking problem

Due to its importance in the domain of warehouse operations, the order picking problem has received significant attention in the past decades [107, 124, 1, 169]. The main body of research on the modelling of order picking problems relies on the mathematical formulation of the Travelling Salesperson Problem (TSP) or the capacitated vehicle routing problem. We divide the existing literature in contributions that focus on exact methods and those that present heuristics.

Almost all of the exact algorithms addressed in the literature for the order picker routing problem, are applied in single-block warehouses. Ratliff and Rosenthal [134] is one of the earliest studies on the optimal order picker routing problem. In this study, the authors introduce a dynamic programming approach for a warehouse layout with only one block (i.e., no cross aisle), which is polynomial in the number of items and aisles. This approach is extended for a two-block warehouse (i.e., one cross aisle) in Roodbergen and De Koster [136] and to a multi-block warehouse (i.e., more than one cross aisle) in Cambazard and Catusse [25]. There exist several extensions to the algorithm proposed by [134], each accounting for slight variations in the problem definition. One of these extensions is the study by Žulj et al. [183]. Here, the authors account for precedence constraint with respect to the picking sequence of the SKUs, based on, e.g., their weight, category, etc. Celik and Süral [31] add additional turn penalties to model that account for the loss of time (speed reduction) each time a picker changes direction within a picking aisle.

The studies by Letchford, Nasiri, and Theis [95], Scholz et al. [147], and Pansart, Catusse, and Cambazard [125] rely on the definition of the Steiner TSP to model the order picker routing problem. In these models, the warehouse is represented by a graph in which the nodes are the union of all SKU storage locations, the depot and all aisle intersections (as these are the main decision points on the route of the picker).

Chabot et al. [33], Irnich, Toth, and Vigo [79], Scholz et al. [147], and Glock and Grosse [61] model the order picking problem as a vehicle routing problem which they solve using branch-and-cut.

For the online order picking problem (i.e., new customer orders arrive over time) Lu et al. [104], Cambazard and Catusse [25], Matusiak et al. [110], and Masae, Glock, and Vichitkunakorn [108] present a dynamic programming formulation.

Apart from these exact methods — which quickly become intractable for increasing instance sizes — a wide range of heuristics and metaheuristics have been developed to solve the order picking problem. According to Masae, Glock, and Grosse [107], (meta)heuristic algorithms account for around 85% of all contributions on the order picker problem.

The best known heuristic algorithms are the *S-shape heuristic*, the *largest gap heuristic* and the *midpoint routing method* [68, 28]. Hybrid extensions of these algorithms have been suggested by Chabot et al. [33], Menéndez et al. [113], Matusiak, De Koster, and Saarinen [109], and Chen, Xu, and Wei [35].

Scholz and Wäscher [148], Theys et al. [160], and Hsieh and Tsai [77] rely on a TSP formulation of the order picking problem which they solve using the well-known *Lin-Kernighan* (LKH) heuristic [101, 71].

Pferschy and Schauer [129] consider different starting and ending points for each pick tour and proposed three heuristics based on different insertion methods combined with a 3-opt local search.

In the majority of the cases, (meta)heuristics have been employed to solve combined problems that integrate order picking with order batching (i.e., group multiple orders together to be picked in a single pick tour, especially useful in e-commerce environments where order sizes are rather small) and batch sequencing (i.e., in which order should the constructed batches be picked such that average or worst response times are minimized). Examples include *particle swarm optimization* [99, 8], *memetic algorithms* [22, 22], *ant colony optimization* [36, 96, 48, 34, 35], *genetic algorithms* [182, 7] and *tabu search* [40].

3.2.2 The Generalized Travelling Salesperson Problem

The Generalized Travelling Salesperson Problem (GTSP) is defined as an extension of the Travelling Salesperson Problem in which the nodes are partitioned into clusters. To serve a cluster it is sufficient that exactly one node of the cluster is visited. The aim is to find a minimum cost / length tour that visits each cluster exactly once.

The GTSP is first introduced by [73]. Since then, the problem has been used to model a wide range of real-life problems [89], such as scheduling problems [17], vehicle routing problems [177], manufacturing problems [156, 76, 90] and telecommunication network design [18].

To solve the GTSP exactly, the problem can be converted into a TSP using dynamic programming, after which it can be solved using dedicated TSP solvers [122]. In the transformed graph, each arc represents the shortest path between each pair of product locations from the original graph, changing the problem into a clustered TSP with a fully connected directed graph, which is afterwards altered into a standard TSP. Such a transformation, however, drastically increases the problem's dimension (in some cases by a factor three or more) [93, 97].

Baniasadi et al. [15] develop a transformation method that is able to convert the Clustered GTSP to a TSP. A Clustered GTSP is defined as a GTSP where in each cluster is further subdivided in multiple sub-clusters that have to be visited consecutively. As the Clustered GTSP reduces to the GTSP with only one sub-cluster per cluster, the transformation method can be applied to turn the GTSP into a TSP instance.

[92, 91] propose a branch-and-bound algorithm to solve the GTSP. In [92] the authors present a first integer programming formulation for the symmetrical GTSP. In Laporte, Mercure, and Nobert [91], the model is extended to the asymmetrical GTSP. In Salman, Ekstedt, and Damaschke [143] a branch-and-bound algorithm is combined with dynamic programming to solve the GTSP with precedence constraints (i.e., some locations have to visited before others) after which they present a comparison between different bounding methods for this problem.

A branch-and-cut strategy is applied by Fischetti, Salazar González, and Toth [56] for the symmetric GTSP. The authors solve a series of LP relaxations while adding valid inequalities to tighten the lower bound using a heuristic algorithm.

[121] develop a *Lagrangian based approach* to transfer the asymmetric GTSP into an asymmetric TSP. Relying on the principles of the Lagrangian relaxation — which removes the flow balancing constraints and adds the corresponding terms to the objective function, making sure that the optimality conditions of the original problem remains — a lower bound to the problem is computed. To find an upper bound, a heuristic algorithm is employed that removes arcs and nodes. This is done by computing the optimal dual solution, the reduced arc costs and their effect on the objective value.

A broad range of heuristic algorithms have been proposed for the GTSP. Similar to what we see for the exact methods, a first group of heuristics relies on a transformation of the GTSP into an asymmetric TSP, e.g. by means of a *Noon-Bean transformation* [20]. Amongst others, the Noon-Bean transformation is used in Helsgaun [72] after which the obtained TSP is solved using the Lin–Kernighan heuristic [101].

Karapetyan and Gutin [83] propose an adaptation of the Lin–Kernighan heuristic for solving the GTSP with non-overlapping clusters directly, by rearranging the path, breaking the path and improving the tour.

Multiple researchers rely on a local-search based algorithm for solving the GTSP by applying the 2-Opt, 3-Opt, and k-Opt operators [100, 23, 180, 86, 67].

Hu and Raidl [78] present a variable neighbourhood search including a generalized 2-opt neighborhood to speed up the search and node exchanges based on the Lin-Kernighan heuristic.

[153] propose a large neighbourhood search based on repeated worst removal and the cheapest insertion of the vertices from and into the tour. In each iteration, one would look for the removal operation (i.e., take out one node in the tour) that reduces the tour length the most. The node is then inserted at the position where it increases the tour length the least.

Other heuristics that have been developed for the GTSP include genetic algorithms [181, 154, 152], memetic algorithms [66, 23], Particle Swarm Optimization [151] and Ant Colony Optimization [180].

Existing models and algorithms developed for the GTSP mainly rely on the implicit assumption that clusters are defined geographically, i.e., clusters do not overlap in the geographical space [54]. Consequently, inter-cluster distances (defined as the distance between two sets of points) can be used as a proxy for the distance between two arbitrary vertices from different clusters and a tour can be constructed first at the cluster level.

Grouping different SKU locations within a warehouse in the same cluster, however, violates this principle and will give rise to an *overlapping graph*. El Krari et al. [54] define overlapping clusters as those that share a geographical space (i.e., when drawing their borders, they find themselves intertwined). To the best of knowledge, only Nalivajevs and Karapetyan [120] consider the order picking problem within a scattered storage policy. The authors developed an instance generator and propose a *conditional Markov chain search* that combines different algorithms to solve the problem at hand. However, the scope of this algorithm is still limited, limiting the instances to single-block warehouse layouts and making them not very representative for today's warehouses.

3.3 A Generalized TSP formulation for order picking under a scattered storage policy

In this section, we formally define the order picking problem under a scattered storage policy. We first introduce the mathematical notation associated with the warehouse layout. Then, we model the problem as a generalized TSP. Finally, we present an improved mathematical formulation for the GTSP.

3.3.1 Representation of the warehouse layout

We assume a traditional multi-parallel aisle (rack and shelf) warehouse that contains a predefined number of pick aisles where SKUs are stored. Additionally, the warehouse contains a given number of intersecting cross aisles that do not contain any SKUs but can be used by the picker to travel efficiently between different pick aisles. As such, the warehouse is divided in a number of blocks, defined as a row of pick aisles between two cross aisles. Moreover, there is a depot located in the front of the warehouse where the picker starts and ends each pick tour (the depot could be seen as the packing area where all picked items are collected and prepared for shipment).

Adopting the common assumptions proposed by Pansart, Catusse, and Cambazard [125], all aisles have equal lengths and are sufficiently narrow such that pickers can pick from both sides while traversing the aisle (i.e., there is no additional delay for changing the pick side). Additionally, pickers can travel through the aisles in any direction and are able to change direction within the aisles if preferred.

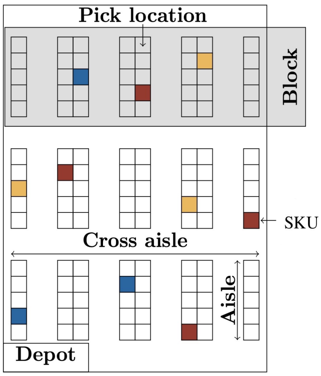
Let $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ be the graph representation of the warehouse, in which $\boldsymbol{V}$ denotes the set of all vertices (SKU locations and the depot) and $\boldsymbol{E}$ is the set of all edges. The depot, included in the set $\boldsymbol{V}$ is denoted by "0". For all $(i, j) \in \boldsymbol{E}$, d_{ij} denotes the distance between storage locations i and j.

Let $\boldsymbol{K}$ be the set of unique SKUs in the warehouse. To represent the storage locations that contain the same SKU, $\boldsymbol{V}$ is partitioned into $|\boldsymbol{K}|+1$ disjoint subsets (one for each SKU and another one that contains solely the depot). These subsets are referred to as *clusters* and denoted by $\boldsymbol{C}_k$ with $k = \{0, 1, \ldots, |\boldsymbol{K}|\})$ — such that $\boldsymbol{V} = \bigcup_k \boldsymbol{C}_k$

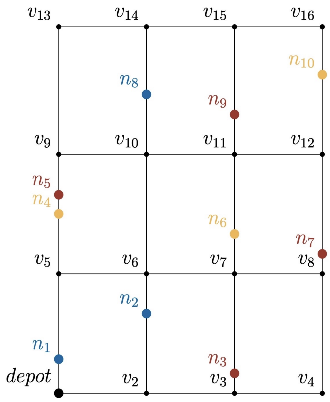
and $\boldsymbol{C}_l \cap \boldsymbol{C}_k = \emptyset \mid l, k \in \{0, 1, \ldots, |\boldsymbol{K}|\}, l \neq k$. Also, given that each SKU can be picked from at least one storage location, $|\boldsymbol{C}_k| \geq 1$.

Based on the pick list containing all the SKUs that should be picked in the current tour, the warehouse representation is reduced by eliminating all SKU locations and the corresponding clusters that should not be visited by the picker. Let $\bar{\boldsymbol{G}} = (\bar{\boldsymbol{V}}, \bar{\boldsymbol{E}})$ be the reduced graph in which $\bar{\boldsymbol{V}}$ is the set of all storage locations of the SKUs that should be picked (including the depot) and $\bar{\boldsymbol{E}}$ the set of all edges connecting the vertices in $\bar{\boldsymbol{V}}$.

Figure 3.1 contains a visual representation of a typical warehouse layout next to it's grid graph representation. Each colour represents a SKU (the vertices in $\boldsymbol{G}$) that should be picked from the warehouse.



(a) Warehouse Layout



(b) Graph Representation

Figure 3.1: *Warehouse layout and graph representation (adapted from Roodbergen [137]). Each colour represents a SKU that should be picked from the warehouse. As multiple locations have the same colour, it is sufficient to visit one location for each colour.*

3.3.2 Definition of the path between two SKU locations

Let (x_i, y_i) be the coordinates of SKU location i, $(i = 1, \ldots, |\bar{\boldsymbol{V}}|)$. Without loss of generality and to simplify notation we assume the direction of x and y align with the cross aisle and pick aisle, respectively. To travel between two SKU locations i and j, the picker will always prefer the shortest path. The calculation of the shortest path length is the same as the previous chapter section 2.3.3.

As we assume a picker will pick each SKU only from a single storage location (i.e., there is always sufficient inventory at each SKU location to satisfy the demand), all d_{ij} for which i and j contain the same SKU (travel between vertices of the same cluster) could be ignored (we do not connect the vertices belonging to the same SKU).

3.3.3 Existing mathematical model formulations

3.3.3.1 DFJ formulation of the GTSP

The first integer linear programming formulation of the GTSP was proposed by [92] using the Dantzig–Fulkerson–Johnson (DFJ) model. They consider the case where at least one node from each cluster must be visited. They proved that for the Euclidean distances case exactly one node from each cluster is visited, using the fact that the distances satisfy the triangle inequality. This inequality also holds in our case, plus the fact that each location is supposed to hold "enough" inventory, thus we can conclude that also in our case exactly one of the locations of each cluster is visited. The Dantzig–Fulkerson–Johnson (DFJ) formulation of [92] is presented below.

In the order picking problem, the nodes refer to the product locations in the warehouse and the edges refer to the shortest path between two nodes. The order picker starts his cycle from the depot and after visiting each cluster exactly once and collecting the products present in the order list, by selecting the shortest route, he returns to the depot. Given the warehouse representation and a list of storage locations to be visited by the picker, let x_{ij} be a binary decision variable denoting whether the edge $(i,j) \in \bar{\boldsymbol{E}}$ is traversed by the picker. Moreover, we will use y_i to model whether vertex $i \in \bar{\boldsymbol{V}}$ is visited by the picker. To ensure that all required SKUs are picked, exactly one y_i variable should be set to 1 in each cluster.

The set of variables and parameters used in mathematical formulations for the GTSP using integer linear programming is described as follows:

$$\min \sum_{(i,j)\in\bar{\boldsymbol{E}}} d_{ij}x_{ij} \tag{3.1}$$

$$\sum_{i\in\boldsymbol{C}_k} y_i = 1 \qquad \forall k \in \bar{\boldsymbol{K}} \tag{3.2}$$

$$\sum_{i\in\bar{\boldsymbol{V}}|i\neq j} x_{ij} = y_j \qquad \forall j \in \bar{\boldsymbol{V}} \tag{3.3}$$

$$\sum_{j\in\bar{\boldsymbol{V}}|j\neq i} x_{ij} = y_i \qquad \forall i \in \bar{\boldsymbol{V}} \tag{3.4}$$

$$\sum_{i\in\boldsymbol{C}_T|i\neq j}\ \sum_{j\in\boldsymbol{C}_T|j\neq i} x_{ij} \leq |\boldsymbol{C}_T| - 1 \qquad \forall \boldsymbol{C}_T \tag{3.5}$$

Table 3.1: *Mathematical notation for the DFJ and MTZ model.*

Sets and indices	
$\boldsymbol{V}$	Set of all the nodes (all storage locations of SKUs) in the graph, $V = \{0, 1, 2, ..., n\}$. The depot node is denoted by 0.
$\bar{\boldsymbol{V}} \subseteq \boldsymbol{V}$	The set of all storage locations of SKUs on the picking list, including the depot.
$\boldsymbol{K}$	The set of all SKUs in the warehouse.
$\bar{\boldsymbol{K}} \subseteq \boldsymbol{K}$	The set of all SKUs on the picking list.
$\boldsymbol{T} \subseteq \bar{\boldsymbol{V}}$	Non-empty subset of nodes in which not all SKUs are represented, i.e., there is a $k \in \{0, \ldots, \bar{\boldsymbol{K}}\} \mid T \cap C_k \mid= 0$.
$\boldsymbol{C}_T$	Set of the SKUs with at least one element and at most $\lvert\bar{\boldsymbol{K}}\rvert$ elements in T in which $1 \leq \lvert C_T\rvert < \lvert\bar{\boldsymbol{K}}\rvert$.
$\boldsymbol{C}_k$	Set of all nodes in the cluster k.
Parameters	
d_{ij}	Length of edge (i, j).
Decision variables	
x_{ij}	Binary variable that equals 1 if edge (i, j) is traversed from i to j; 0 otherwise.
y_i	Binary variable that equals 1 if vertex i is visited and 0 otherwise.
u_p	order of visiting cluster p in the tour for MTZ subtour elimination constraint
w_{pq}	Binary variable that equals 1, if the order-picker visits a node of SKU q immediately after visiting a node from SKU p and 0 otherwise

The objective function (1) minimizes the total distance travelled by the pickers. Constraints (2) ensure that every SKU must be visited exactly once. The flow constraints are indicated by constraints (3) and (4). These constraints ensure that if an item is picked in a given tour, one incoming and one outgoing edge to this node from other nodes should exist. Constraints (5) are the subtour elimination constraints.

3.3.3.2 MTZ formulation of GTSP

The second ILP formulation considered in this study is the GTSP formulation proposed by [82] and improved by [26] using Miller–Tucker–Zemlin (MTZ) subtour elimination constraints. The mathematical formulation is the following:

$$\min \sum_{(i,j)\in \bar{\boldsymbol{E}}} d_{ij}x_{ij} \tag{3.6}$$

$$w_{pq} = \sum_{i\in \boldsymbol{C}_p}\sum_{j\in \boldsymbol{C}_q} x_{ij} \qquad \forall p,q \in \bar{\boldsymbol{K}} \mid p \neq q \tag{3.7}$$

$$\sum_{p|p\neq q} w_{pq} = 1 \qquad \forall q \in \bar{\boldsymbol{K}} \tag{3.8}$$

$$\sum_{q|q\neq p} w_{pq} = 1 \qquad \forall p \in \bar{\boldsymbol{K}} \tag{3.9}$$

$$\sum_{j\in \bar{\boldsymbol{V}}\setminus\{i\}} x_{ji} - \sum_{j\in \bar{\boldsymbol{V}}\setminus\{i\}} x_{ij} = 0 \qquad \forall i \in \bar{\boldsymbol{V}} \tag{3.10}$$

$$u_p - u_q + |\bar{\boldsymbol{K}}|w_{pq} \leq |\bar{\boldsymbol{K}}|-1 \qquad \forall p,q \in \bar{\boldsymbol{K}} \mid p \neq q \tag{3.11}$$

$$2 \leq u_p \leq |\bar{\boldsymbol{K}}|+1 \qquad \forall p \in \bar{\boldsymbol{K}} \mid p \neq 1; u_1 = 1 \tag{3.12}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in \bar{\boldsymbol{V}} \tag{3.13}$$

The objective function (3.6) is minimizing total distance traveled by the order picker. Constraints(3.7) are the definition of the auxiliary variables, w_{pq}, in terms of the defined decision variables x_{ij}. Constraints (3.8) and (3.9) ensure that every SKU must be visited exactly once by ensuring that the degree of the incoming and outgoing arcs from each SKU is equal to 1, respectively. Constraints (3.10) state that entering flow to every node should be equal to exiting flow (flow balance constraints). Constraints (3.11 and 3.12) are the subtour elimination constraints, which represent the visiting order for all SKUs.

3.3.4 Reflection on existing formulations

In the DFJ formulation, the computational time goes up easily as the number of constraints grows exponentially in the number of nodes in the graph. More specifically, the number of subtour elimination constraints equals $(2^{\bar{K}} - \bar{K} - 1)$, where $\bar{K}$ is the number of SKUs. This implies that solving this model must be done with a separation routine on the subtour elimination constraints, which is time-consuming.

[82] show that the MTZ formulation contains O(n^2) binary variables and O(n^2) constraints. As such, the number of variables and constraints are polynomial in the size of the problem. This formulation, however, makes use of three types of variables: one based on the nodes (all SKU locations), one based on the clusters (subsets of SKU locations that contain the same SKU), and one based on the arcs.

3.3.5 A new mathematical formulation for the GTSP

In this section, we present a new mathematical formulation for the GTSP that combines the beneficial characteristics of the above-mentioned DFJ and MTZ formulations. This new formulation is based on decision variables at the storage location level. For the subtour elimination constraints, we rely on the MTZ approach.

Table 3.3: *Summary of all mathematical notation.*

Sets	
$\boldsymbol{V}$	The set of all the vertices in the graph (complete warehouse).
$\bar{\boldsymbol{V}} \subseteq \boldsymbol{V}$	The set of all storage locations of SKUs on the picking list, including the depot.
$\boldsymbol{C}_k$	The set of all vertices in cluster k.
$\boldsymbol{K}$	The set of all SKUs in the warehouse.
$\bar{\boldsymbol{K}} \subseteq \boldsymbol{K}$	The set of all SKUs on the picking list.
Parameters	
d_{ij}	Length of edge (i, j).
Decision variables	
x_{ij}	Binary variable that equals 1 if edge (i, j) is traversed from i to j; 0 otherwise.
y_i	Binary variable that equals 1 if vertex i is visited and 0 otherwise.
u_i	The position of vertex i in the tour.

$$\min \sum_{(i,j)\in\bar{\boldsymbol{E}}} d_{ij}x_{ij} \tag{3.14}$$

$$\sum_{i\in\boldsymbol{C}_k} y_i = 1 \qquad \forall k \in \bar{\boldsymbol{K}} \tag{3.15}$$

$$\sum_{i\in\bar{\boldsymbol{V}}|i\neq j} x_{ij} = y_j \qquad \forall j \in \bar{\boldsymbol{V}} \tag{3.16}$$

$$\sum_{j\in\bar{\boldsymbol{V}}|j\neq i} x_{ij} = y_i \qquad \forall i \in \bar{\boldsymbol{V}} \tag{3.17}$$

$$u_0 = 1 \tag{3.18}$$

$$u_i - u_j + |\bar{\boldsymbol{K}}|x_{ij} \leq |\bar{\boldsymbol{K}}|-1 \qquad \forall i, j \in \bar{\boldsymbol{V}}\backslash\{0\} \mid i \neq j \tag{3.19}$$

$$2 \leq u_i \leq |\bar{\boldsymbol{K}}|+1 \qquad \forall i \in \bar{\boldsymbol{V}}\backslash\{0\} \tag{3.20}$$

$$u_i \in \mathbb{N} \qquad \forall i \in \bar{\boldsymbol{V}} \tag{3.21}$$

$$x_{ij}, y_j \in \{0, 1\} \qquad \forall i, j \in \bar{\boldsymbol{V}} \tag{3.22}$$

The objective function (3.14) minimizes the total distance travelled by the picker. Constraints (3.15) ensure that each SKU on the pick list is picked exactly once. Constraints (3.16) and (3.17) set the incoming and outgoing edges for each visited cluster. In constraints (3.18) the depot is set as the first vertex in the pick tour. The positions of all other vertices in the tour are set via constraints (3.19), after which the domain of these positions is bounded by the number of clusters that will be visited in the complete pick tour in constraints (3.20). Finally, the domains of the decision variables are managed by constraints (3.21) and (3.22).

3.4 Guided Local Search algorithm

To solve the order picking problem under a scattered storage policy, we develop a strong heuristic algorithm based on the Variable Neighbourhood Search (VNS), which is one of the most recent meta-heuristics for local search, embedded in a Guided Local Search (GLS) framework. The GLS algorithm sits on top of a local search and makes use of search related information in order to change its behavior and guide the local search [165]. The GLS adds a set of penalty terms to the cost function of the problem, to help local search algorithms escape from local minima. Whenever local search gets caught in a local optimum, the penalties and objective function are modified and local search is called again to optimize the modified cost function.

In this variant of local search, problem-specific features of the solution are considered directly in the objective function, thereby 'guiding' the search to solutions that possess most features of high quality solutions.

3.4.1 Overview of our GLS procedure

In this section, we provide a brief overview of the different components within our GLS algorithm. In the following sections, we will elaborate on each of these in more details. The flow diagram of our GLS algorithm is provided in Figure 3.2. In figure 3.2, r is the set of shaking phase operators, for $r = 1, ..., r_{max}$, and l is the set of local search moves, for $l = 1, ..., l_{max}$. M is the number of features. To avoid the complexity in the algorithm, the VNS step (line 10) is illustrated in pink box in more details in figure 3.2. For a better understanding, the pseudo-code of VNS is presented in algorithm 1.

First, an initial solution is generated by means of a constructive heuristic. As we only accept feasible solutions, this initial solution will be a tour that starts and ends at the depot and visits exactly one storage location for each required SKU.

Then, we evaluate the initial solution based on its *augmented objective function*. The augmented objective function is a combination of the actual objective function (in our case the minimization of the total distance travelled) with a series of penalty terms. These penalty terms focus on problem-specific features that are likely not appearing

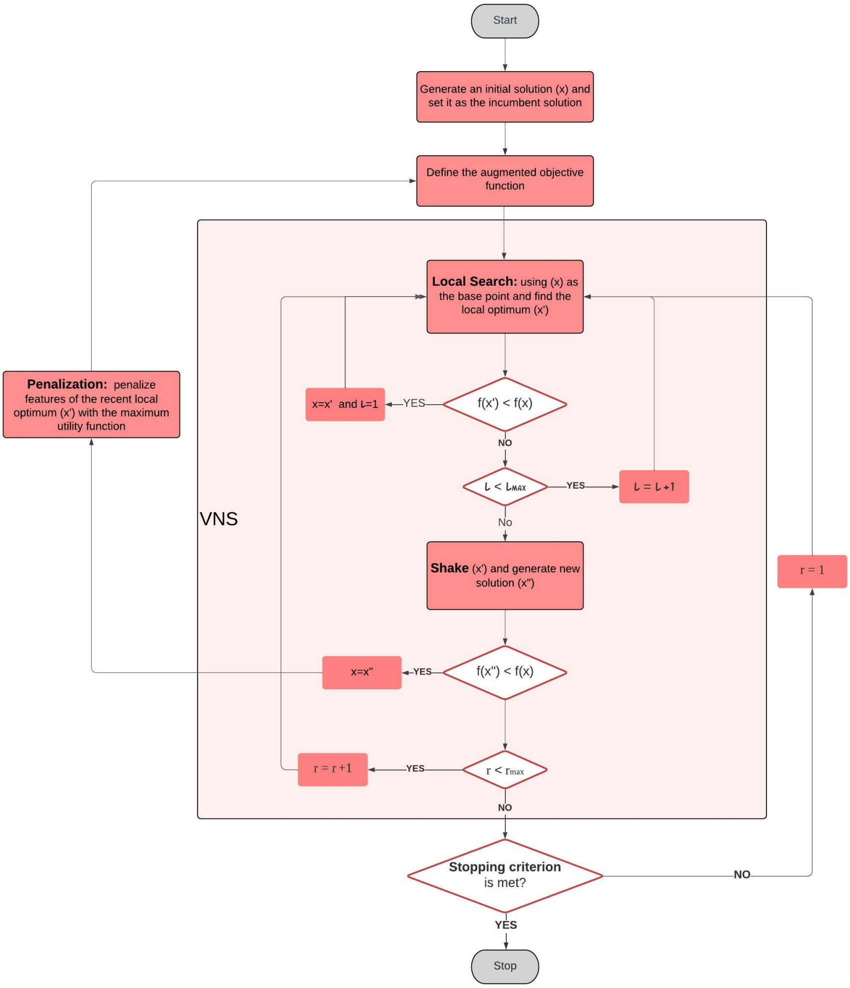


Figure 3.2: Overview of the Guided Local Search algorithm.

Algorithm 1 VNS Algorithm

$x \leftarrow$ initial solution made by a Construction Method; ▷ Set x as the incumbent solution
while Stopping Criterion **do**
  $r \leftarrow 1$;
  while $r \leq r_{\max}$ **do**
    {Shaking}: Generate a starting point x' at random from $N_r(x)$;
    {Local search}
    Set $\ell \leftarrow 1$
    while $\ell \leq \ell_{\max}$ **do**
      Explore the $N_\ell(x')$ and find the best neighbor, x'' ;
      {Move or not}:
      if $f(x'') < f(x')$ **then**
        set $x' \leftarrow x''$ and $\ell \leftarrow 1$;
      else set $\ell \leftarrow \ell + 1$;
      end if
    end while
    {Improve or not}
    if $f(x'') < f(x)$ **then**
      $(x \leftarrow x'')$ and $(r \leftarrow 1)$;
    else set $r \leftarrow r + 1$;
    end if
  end while
end while
$x \leftarrow$ best solution found with respect to objective function g;
return x

in the optimal solution (e.g., very long travel distance between two consecutive picks). By penalizing these undesirable solution features, we help the algorithm to converge towards the more promising regions of the solution space faster. Moreover, the penalties are updated dynamically to help the local search heuristic to escape local optima (e.g., by suddenly allowing solutions that score less high on the included criteria) or to focus on high quality regions (e.g., by punishing certain criteria more) [173].

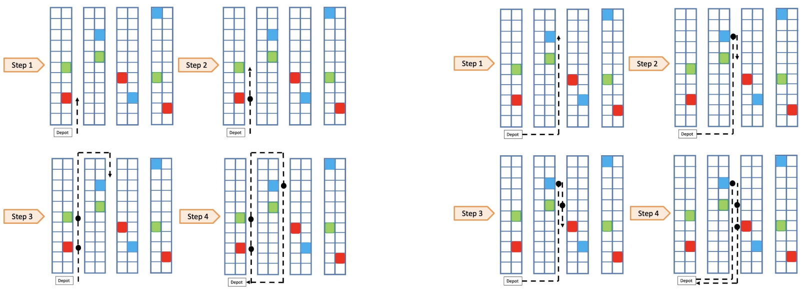
In the actual local search phase, we improve the current solution by exploring a series of local search neighbourhoods sequentially. As such, we make use of the principles of *Variable Neighbourhood Search*. If no improvements can be found (i.e., the algorithm is stuck in a local optimum), a shaking phase – often referred to as a *perturbation* – helps to reach different regions of the solution space, after which the algorithm continues its search.

3.4.2 Step 1: Generate an initial solution

The aim of this step is to generate a feasible solution to the order picking problem under a scattered storage policy. To achieve this, we propose the following two approaches: a greedy nearest neighbour heuristic and a farthest insertion algorithm. The performance of both methods on the final solution is assessed (see Section 3.5) but no significant difference in final results (i.e., after the local search) could be found. Thus, since both of the constructive algorithms find the solution in less than one second, we run both algorithms, we select the best generated solution among them and use it as the initial solution for our GLS heuristic.

The first method, a *greedy nearest neighbour heuristic* is an adaptation of the nearest neighbour heuristic for the TSP. The picker starts at the depot after which the storage location that contains an SKU on the pick list and is closest to the current location is visited. Each time a storage location is added to the pick tour, all storage locations from the same cluster are removed from the graph. This procedure is repeated iteratively until all required SKUs have been picked. Then the picker moves from its current location back to the depot. As all distances in the graph are known, the procedure is very fast. However, due to the myopic decisions (i.e., we only consider a decision on the next location to visit), inefficient connections towards the end of the route are likely.

Second, we propose a *farthest insertion algorithm*, which works as follows: in the first step, we find for each SKU on the pick list the storage location that is closest to the depot. Then, among these locations we add the one that is farthest away from the depot to the pick tour. Knowing that we have to pick each SKU on the pick list, we know that for sure we need to travel up to this point into the warehouse (i.e., the nearest storage location of the SKU that is furthest away from the depot). For each next SKU, we perform a cheapest insertion (i.e., the storage location that increases the length of the pick tour the least is included into the partial tour. We continue until all required SKUs have been visited by the picker. Figures 3.3a and 3.3b illustrate an example of the greedy and farthest insertion constructive solution respectively for the case where 3 clusters of products with 3 items (nodes) each exist.



***(a)** Greedy nearest neighbour heuristic.* ***(b)** Farthest insertion algorithm.*

***Figure 3.3:** Constructive algorithms to generate an initial solution.*

3.4.3 Step 2: Define the augmented objective function

Following the definitions from Alsheddy et al. [5], let $\boldsymbol{F}$ be set of solution features by which the quality of a solution can be assessed. For each feature $i \in \boldsymbol{F}$, let I_i be a binary variable indicating whether the feature is present in the current solution or not (also denoted as the *indicator function*). Let p_i record the number of times that feature i has been penalised (appeared in the local minima) and initially it is set to zero. Finally, we define λ as an overall weight factor for penalty dedicated to the features in the objective function. As such, λ balances the importance of the penalty factors over the main objective function (i.e., the minimization of the total distance travelled) and, thereby, controls the degree of guidance within the GLS. The value of λ is defined with parameter tuning (the value of λ depends on another parameter (α which will be explained later). Furthermore, let $c = \{c_1, c_2, \dots, c_{|\boldsymbol{F}|}\}$ be a cost vector, in which c_i denotes the cost of feature i and will be defined later.

Denoting the main objective function (i.e., minimizing the total distance travelled by the order picker) by $g(s)$, the augmented cost function is given by

$$h(s) = g(s) + \lambda \sum_{i \in \boldsymbol{F}} p_i I_i(s) c_i \tag{3.23}$$

The objective within the GLS algorithm is to minimize the augmented cost function $h(s)$.

Each time a local minimum s^* is found by the algorithm (and thus the augmented cost function cannot be optimised further), we will evaluate the current solution based on its solution features. This evaluation is done based on a so-called utility function $util(s^*, f_i)$ that scores the solution s^* on each feature i. Then the algorithm penalizes the maximum utility of each features. For instance, if our feature is the length of the edges in the tour, the algorithm calculates the utility of this feature and penalizes the longest edge in the current tour. Thus, for a local minimum, we calculate the maximum utility of every feature and penalize it in our augmented objective function. The general formulation for the utility function is as follows:

$$util(s^*, i) = I_i(s^*) \frac{c_i}{(1 + p_i)} \tag{3.24}$$

If a feature is not present in s^* (denoted by the indicator function $I_i(s^*) = 0$), then the utility of penalizing it equals to 0. For a feature that is present ($I_i(s^*) = 1$), its cost c_i will be computed. If a feature is penalized multiple iterations in a row, its penalty parameter p_i (which works as a counter) increases which will dampen the importance of the feature's utility which, at its turn, inserts more diversification on the search (i.e., other features will become more important in the augmented cost function).

To guide our local search to the more promising solutions, we define the following features:

1. **The maximum number of times that each picker-aisle is visited.**
   This feature tries to force the algorithm to pick all the items needed in the same aisle in one visit and avoids visiting an aisle more than twice (we know that in optimal solution, each aisle should be visited at most once in each direction). Thus, the algorithm chooses the aisle with the maximum number of visits (maximum utility) and penalizes it. Let T^a be the number of times that aisle $a \in A$ is visited (A is the set of aisles) in our current solution s^*, then the utility of this aisle is given by:
   $$util(s^*, 1) = \max_a \left[I_1^a(s^*) \frac{T^a}{p_1^a + 1} \right] \tag{3.25}$$
   As part of our augmented objective function, we will punish the aisle with the largest utility. p_1^a is the penalty counter for aisle a.

2. **The maximum number of times that each cross-aisle is visited.**
   Similar to feature one, here we try to penalize the number of times that each cross-aisle is visited, so that we reduce the extra movements of an order picker in the warehouse. Let T^c be the number of times that cross-aisle $c \in C$ is visited (C is the set of cross-aisles) in our current solution s^*, then the utility of this

cross-aisle is given by:

$$util(s^*, 2) = \max_c \left[I_2^c(s^*) \frac{T^c}{p_2^c + 1} \right] \tag{3.26}$$

As part of our augmented objective function, we will punish the cross-aisle with the largest utility. p_2^c is the penalty counter for cross-aisle c.

3. **The maximum number of SKUs picked from each aisle.**

   The idea is that we would like to pick all items, while having to visit as few aisles as possible. Unlike the first two penalties, the cost associated with this feature is not as clear. The algorithm aims to visit the minimum number of aisles, which contain items from a large variety of clusters. Hence, the aim is to traverse those aisles which contain a large set of items from different clusters. Therefore, we wish to penalize the aisles which have low utilization (visited while picking the least number of SKUs in it). Considering that in the optimal case, we expect that in an aisle visit, an order-picker collects as many products as possible, in order to reduce the number of times other aisles are visited. To guide the search mechanism based on this feature, let N_a be the number of SKUs picked in aisle a in our current solution s^*, then the cost of this feature is defined as $\left(\frac{1}{N_a}\right)$. As a result, the utility of this feature is given by:

$$util(s^*, 3) = \max_a \left[I_3^a(s^*) \frac{1}{N_a(p_3^a + 1)} \right] \tag{3.27}$$

   As part of our augmented objective function, we will punish the aisle with the largest utility. p_3^a is the penalty counter for aisle a.

4. **The longest edges.** Our last important feature to be penalized is the longest edge travelled in the warehouse. If the length of an edge ij which is the shortest path connecting node i to node j in our solution is too high, it can be a sign of non-optimality since it may be the case that the order-picker has visited some aisles without picking any item on his way, or maybe the destination node of the edge could be visited by another closer node. Intuitively, we do not want long edges but cannot exclude them at the beginning of the search procedure, as they may be part of the optimal solution. Thus, we penalize them during the search if they appear in local optima instead of disregarding them.

   The utility function of this feature is as follows:

$$util(s^*, 4) = \max_{ij} \left[I_4^{ij}(s^*) \frac{d_{ij} x_{ij}}{p_4^{ij} + 1} \right] \tag{3.28}$$

where x_{ij} is a binary variable taking value 1 if order picker visits node j immediately after visiting node i and d_{ij} denotes the length of the edge. A natural choice for the cost associated with these solution features is the length of the analogous edge.

3.4.4 Step 3: Improvement by means of local search

Variable Neighbourhood Search

To improve the quality of the initial solution, we make use of variable neighbourhood search. Having defined multiple local search operators, the algorithm changes the operator as soon as no further improvement can be found in the current neighbourhood (i.e., the search is stuck in a local optimum). The following neighbourhoods are checked consecutively within our algorithm (see figure 3.4):

- 2-OPT. For a given set of two arcs in a single route that construct a crisscross (cross each other), this move substitutes them with two new arcs by reversing the sequence of the nodes visited in between.
- 3-OPT. Remove three arcs and interchange their position in the itinerary.
- INTRA-SWAP. This move selects two random nodes (clusters) in our current solution (route) and swaps the nodes of these positions in the current route.
- INSERT. This move selects two random positions of nodes in our current solution (route) and inserts one of these randomly chosen elements in front of the second element.
- REVERSE. This move selects two random positions of nodes in our current solution (route) and reverses the local path between these two randomly chosen elements.

Shaking phase

Once a local optimum has been reached, the algorithm makes use of a shaking procedure to escape from it. The shaking procedure, inspired by the work of Sengupta, Mariescu-Istodor, and Fränti [150] and Tuononen [166], executes random moves from either of the two following operators.

- INTER-SWAP. This move selects one random node (cluster) in our current solution (route) and replaces another node of the same cluster (which currently is not part of the solution) with the existing node of that cluster (see Figure 3.5).

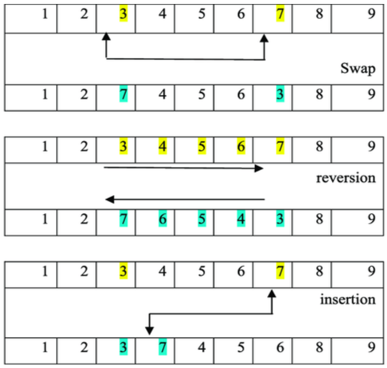


Figure 3.4: *Moves considered in VNS algorithm from [119]*

- DOUBLE-BRIDGE. This move consists of a sequence of two disconnected 2-exchange moves. In the first exchange, the algorithm removes two random edges from the current tour and links their endpoints by adding new edges, resulting in two sub-tours. The second exchange removes two other edges, one from each sub-tour and reconnects the two parts by creating a bridge and making a feasible tour (see Figure 3.6).

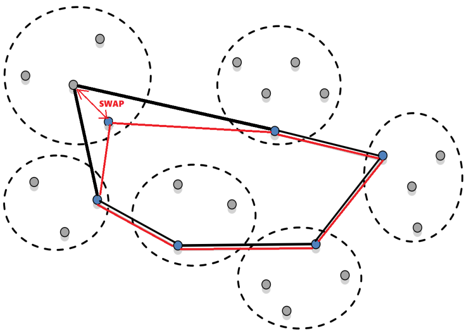


Figure 3.5: *Visualisation of the* INTER-SWAP *move.*

3.4.5 Stopping criterion

The algorithm stops as soon as a maximum number of iterations of the VNS without improvement has been performed (which we fix to 1000) or if the running time ex-

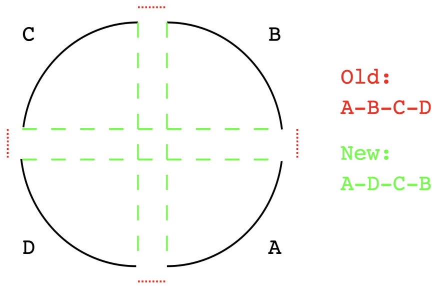


***Figure 3.6:** Visualisation of the* DOUBLE-BRIDGE *move.*

ceeds 60 minutes.

3.4.6 GLS pseudo-code

The pseudo-code of our GLS algorithm is provided in Algorithm 2.

Algorithm 2 Pseudo-code for our Guided Local Search algorithm (p,g,λ, $(I_1, ..., I_m), (c_1, ..., c_m)M)$).

$k \leftarrow 0$; ▷ iteration counter
$x \leftarrow$ initial solution made by a Construction Method;
{ set all penalties to 0}
for $i \leftarrow 1$ until M **do**
    $p_i \leftarrow 0$;
end for
{define the augmented objective function }
update the augmented objective value and set $h(x) \leftarrow g(x) + \lambda \sum_{i \in M} p_i I_i c_i$
while Stopping Criterion **do**
    $x_{k+1} \leftarrow$ VNS (x_k, h);
    {compute the utility of features }
    {penalize features with maximum utility}
    for All i such that $util_i$ is maximum **do**
        $p_i \leftarrow p_i + 1$;
    end for
    $k \leftarrow k + 1$;
end while
$x^* \leftarrow$ best solution found with respect to objective function g;
return x^*

3.5 Model implementation and numerical results

3.5.1 The instances

We perform computational experiments by generating a set of random instances, based on the instance generation procedure described in Theys et al. [159]. [1]

The benchmark used in our study is the one from [125] and [159] consisting of 27 different scenarios: three different number of aisles (5,15,60), three different number of cross aisles (3,6,11) and three different number of products in the order (15,60,240). As previously mentioned, there are no papers in the literature considering the GTSP in multi parallel aisle warehouses and the order picking problem.

3.5.2 Some details on implementation

All algorithms presented in this chapter are implemented in Java, and the ILP formulations are solved with IBM CPLEX 12.1.0 with default parameters. The time limit for the exact algorithms has been set to 3600 seconds. Testing has been carried out on a 11th Gen Intel(R)Core i5-1135G7@2.0GHz and 8GB of RAM. It will be shown that high-quality solutions can be generated for real size instances of the order picker routing problem within a reasonable time. Please note that our instances are not exactly the same than those solved by [148] but were generated with the same parameters. It is worth mentioning that in all the result tables, GLS refers to the case where VNS and GLS are combined and VNS refers to the single VNS heuristic without any penalizing and augmented objective function. This is done in order to see the effect of "guiding" on the solution and the computation times. Moreover, for each instance, we run the heuristic algorithms (VNS and GLS) 10 times and report the mean values in the tables.

3.5.3 Parameter tuning

Another important part of our sensitivity analysis is the parameter tuning in order to find the best values for the penalty coefficients (λ) in our GLS algorithm. Recall that the λ parameters control the degree to which the respective penalties influence the search procedure. As it has been previously stated, these parameters present a trade-off between exploration and exploitation of the search space. Hence, the choice of the values associated with these parameters will impact the efficiency of the search procedure. Since the effectiveness of exploration and exploitation is highly dependent on the landscape of the search space, the choice of the λ parameters will, in general, be instance specific. Based on computational experiments for several problems, [174] observed that the basis of obtaining good values for these parameters can be found by

[1] More information about the instances and instance generator can be found via https://homepages.dcc.ufmg.br/~arbex/orderpicking.html.

dividing the objective value within a local optimum by the number of solution features present in this solution. In another words, λ is computed dynamically after finding the first local optimum and before penalizing the features for first time. Having an α parameter which is calculated by sensitivity analysis, λ is calculated by

$$\lambda = \frac{\alpha g(s^*)}{|F_{s^*}|} \tag{3.29}$$

in which s^* and F_{s^*} denote the local optimum and the features present in the solution. The value of the α parameter should be between 0 and 1, using the information from [173] since we are using the same 2-opt and 3-opt operators in our VNS heuristic. To optimize performance on a built model, parameter tuning is necessary for any algorithm. To do so, we add each penalising feature to the objective function separately and then solve the model (run the algorithm) for several values of α and find the best point where our total cost is minimized. Figure 3.7 illustrates the optimal value for our α parameter.

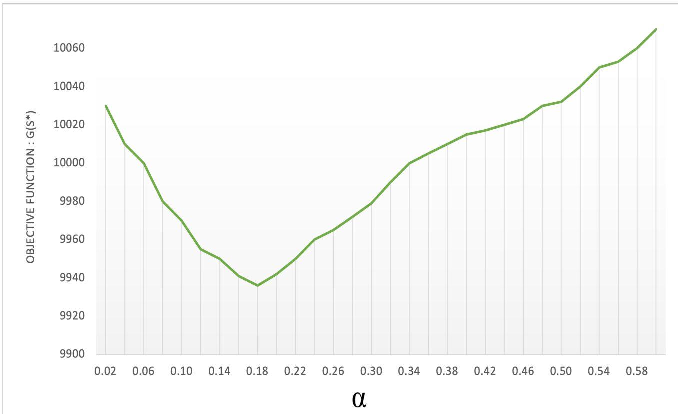


Figure 3.7: *Sensitivity analysis of GLS algorithm for optimal tuning of* α

3.5.4 Comparison between state-of-the-art formulations on traditional problem

We first assume that in our GTSP, each cluster consists of only one node, changing the problem to normal TSP, and then we solved our exact and heuristic algorithms to have a better comparison and to examine the efficiency of our proposed algorithm. In table 3.5, we solve the instances on the multi-block warehouse and each cluster containing only one location (TSP). As shown in this figure, for all the instances of different sizes, our model has the optimal solution in less than one minute. The calculation times are based on CPLEX solver time. It is illustrated in this figure that our proposed MILP(MILP proposed) gives better solutions in comparison with the MILP in the literature using MTZ formulation (given in column 'MILP old') in a shorter running time. In this table 'O' means Optimal solution, 'F' is used to show Feasible solution and NS means no solution. In this table, MILP best cost refers to the the best objective value among the first two columns.

Both of our constructive algorithms (greedy and Farthest-Insertion) give us the initial solution in less than one second. Furthermore, their solution quality is not comparable since non of them is outperforming the other one in the solution quality. For some instances, the greedy algorithm gives better solutions and for the other ones, the Farthest-Insertion. No pattern for their solution quality based on different instance sizes has been found. Thus we run both algorithm and since they are both very fast, we select the best generated solution among them and use it as the initial solution for our GLS heuristic. To have an analysis over the efficiency of the GLS and how much it improves the VNS without penalties, we separate these two parts and solve the instances with both of them (VNS refers to the case where there is no penalising and GLS refers to the VNS embedded in GLS having penalty functions). Since in this table, we consider only one item inside each cluster (TSP), the instance sizes are smaller than the other sets and our MILP model is giving optimal solution for most of the instances. Therefore, the improvement of the mathematical model by VNS is on average 6.5%. Also the average improvement of VNS by adding GLS is 2%. This number gets higher if the instance sizes (items in each cluster) increase.

It is worth mentioning, that our GLS heuristic is capable of finding the optimal solutions for most of the instances along with our proposed MILP, but in a much shorter running time (less than 3.5 seconds for all instances). The other non-optimal instances, are having a huge negative gap with our exact solution, which shows the improvement in our heuristic solution. For bigger instances, the heuristic stops by the termination criteria regarding no improvements for 1000 iterations.

3.5.5 Performance analysis for increasing cluster size

In this section, we examine the performance of our exact and heuristic algorithms on 4 different cluster sizes (2,5,10,20) and solve our instances based on these cluster sizes. In tables 3.6-3.9, the numerical results for the cluster sizes 2,5,10 and 20 are reported respectively. The "MILP model old" referes to the results from the old MTZ formulation in the literature solved and implemented on our computer with CPLEX 22.1. As it is shown in these tables, with the increase in cluster sizes and accordingly the instance sizes, our proposed MILP is not able to give us a feasible solution within the time limit of one hour. However, our heuristic algorithm is able to solve the problem and give us a good solution in a very short time (for our biggest instance with 4800 nodes, the running time is 52.5 seconds which makes it remarkably fast). For the cluster size 1,2 and 5, our proposed MILP can provide optimal or feasible solutions, but for the larger instances, the model is not able to give any solutions within 60 minutes. The other notable point in these tables is the comparison between our proposed MILP and the existing MILP in the literature. On average, the running time of our proposed MILP is 20% less than the other MILP in the literature, and the quality of the solutions generated by our MILP is on average 15% better than the existing MILP.

Comparing the solutions before and after implementing the guided local search algorithm (we call them GLS and VNS (which does not contain guided local search)), we can see that implementing GLS and penalizing the features of the solutions has an improvement of 15% on average; and this percentage is higher for bigger instances than for small instances with 2 products in each cluster, for which the average improvement is 4%. The comparison between objective functions of our proposed MILP and VNS and GLS for instances with cluster size 2,10 and 20 are shown in figures 3.8-3.10.

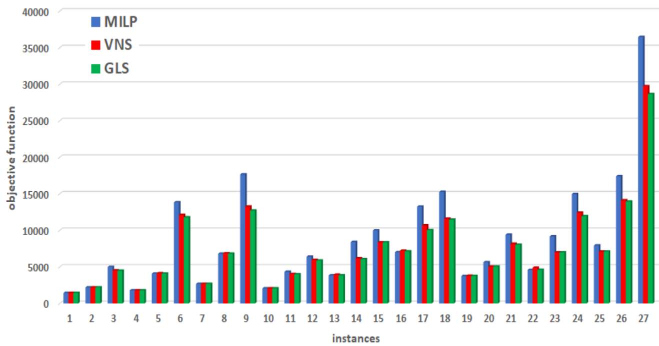


Figure 3.8: *Comparing solution methods for instances with cluster size=2*

In table 3.10, we analyse the mean and the standard deviation of the solutions for different instances, and in figure 3.11 these values are illustrated by the boxplot. In our heuristic algorithm, we solve each large-scale instance (each cluster containing 20 products) 10 times and calculate the mean and standard deviations of the solutions in order to analyse the stability of our algorithm. Thus, the observations hold in each boxplot is the 10 results obtained from the repetitions of the application of the algorithm. As is shown in figure 3.11, the size of the boxes for different instances in a VNS algorithm is larger than the size of boxes in GLS, which means that the standard deviation of the solutions generated by GLS is lower than the ones generated by a VNS

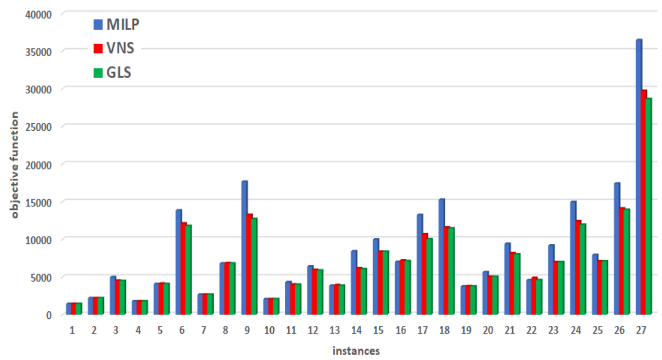


Figure 3.9: *Comparing solution methods for instances with cluster size=10*

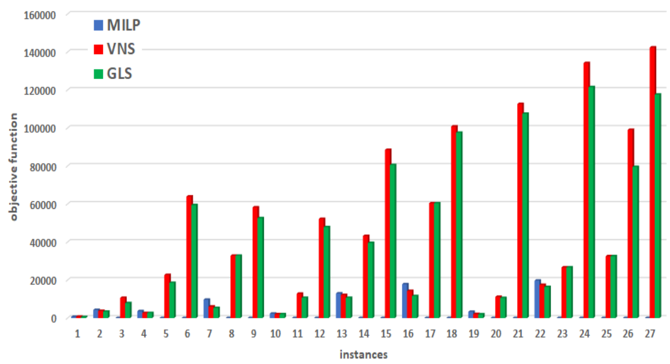


Figure 3.10: *Comparing solution methods for instances with cluster size=20*

algorithm. In other words, a GLS algorithm is much more stable and robust compared to the VNS.

Furthermore, the percentage of improvements of GLS implemented on a VNS algorithm for cluster sizes 2,10 and 20 is illustrated in figures 3.12-3.14. For the smaller cluster size (2), the average improvement achieved by GLS is 4% which is much lower than the percentage of improvements in larger instances (cluster size 20) for which this value is 15%. The reason for this is that in the smaller instances, our VNS algorithm is also able to find optimal or close to optimal solutions. Therefore, a GLS algorithm does not add to much to the VNS algorithm. However, in bigger instances, the solutions found by GLS have much better quality and the running times of the GLS is much lower than the VNS or MILP. In figures 3.15-3.18 the comparison between the running time of VNS and GLS algorithms are illustrated for different cluster sizes. These charts show that in the smaller instances (cluster size 2 and 5) the running time of GLS is on average 80% less than the running time of VNS, and this value drops to 50% for larger instances with 10 and 20 products in each cluster. These numbers show the efficiency of our proposed GLS algorithm and its significant improvements both in run time and the quality of the solutions.

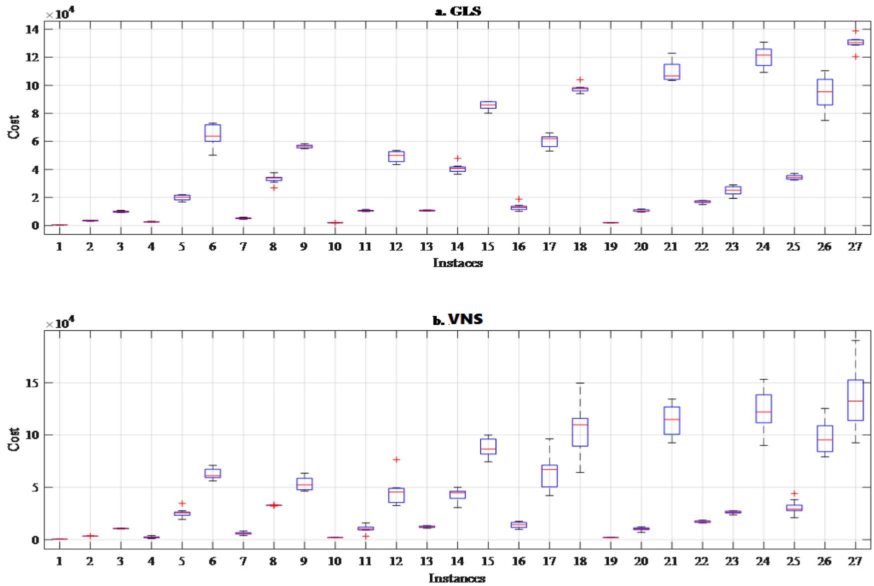


Figure 3.11: *Boxplots of VNS and GLS to compare the stability of algorithms*

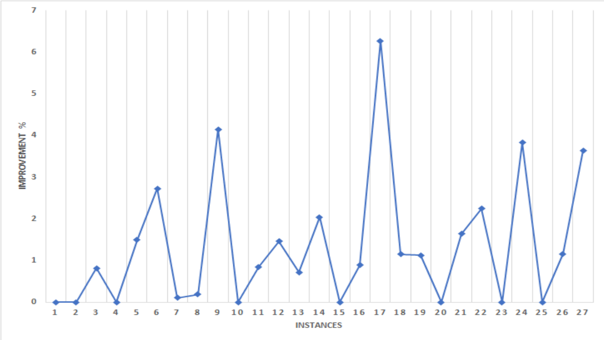


Figure 3.12: *Relative improvement of VNS by applying GLS in case of cluster size=2*

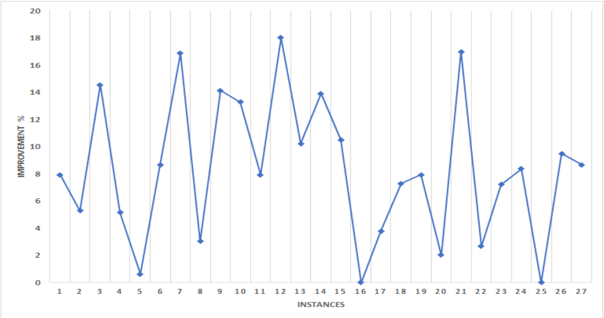


Figure 3.13: *Relative improvement of VNS by applying GLS in case of cluster size=10*

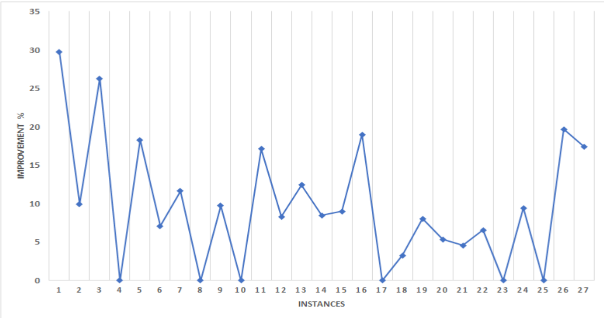


Figure 3.14: *Relative improvement of VNS by applying GLS in case of cluster size=20*

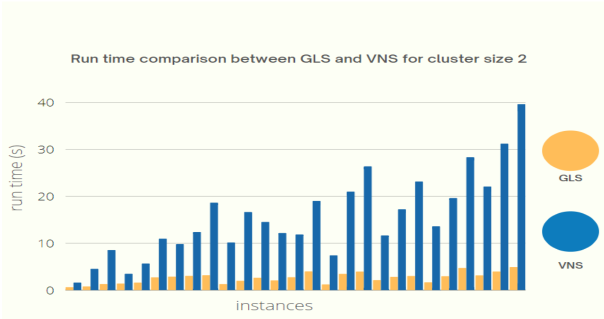


Figure 3.15: *Comparison between the running time of VNS & GLS in case of cluster size=2*

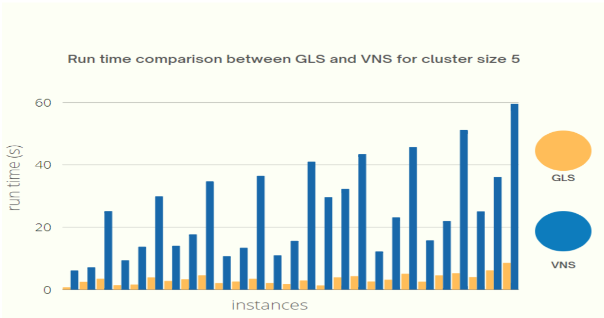


Figure 3.16: *Comparison between the running time of VNS & GLS for cluster size=5*

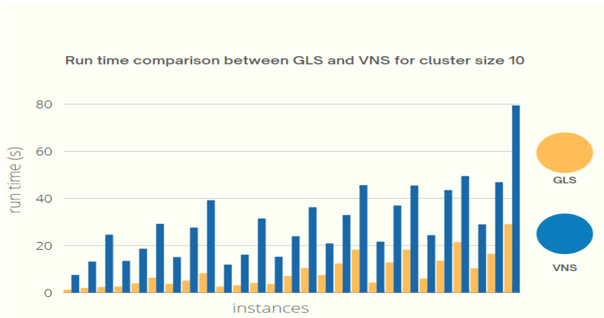


Figure 3.17: *Comparison between the running time of VNS & GLS for cluster size=10*

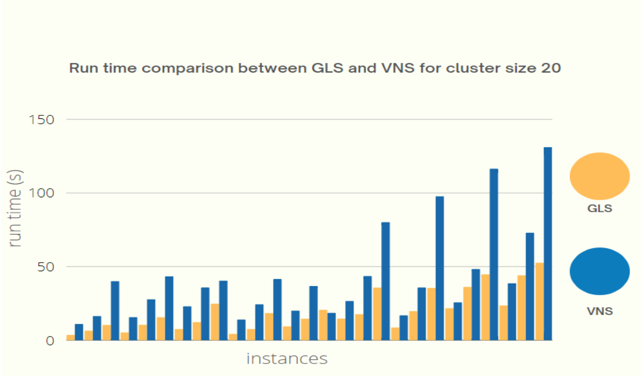


Figure 3.18: *Comparison between the running time of VNS & GLS for cluster size=20*

3.5.6 Warehouse Layout Comparison

In this section, we examine the effects of different warehouse sizes on the proposed algorithm. To do so, we consider instances with 20,60 and 100 products and 5,10,15 and 20 clusters. The average computational running times for all problem instances are plotted in figure 3.19. The figure is separated into sub-figures for the instances related to the different cluster sizes. Furthermore, the nine different warehouse configurations are sorted along the x-axis according to the increasing number of total aisles contained in the warehouse. For example, the label k5h3 refers to the configuration with five aisles and three cross aisles, which is the smallest configuration considered in this chapter. In this graph, n is considered as the number of products in the instance.

In this figure, it is shown that the computational performance across all configurations with twenty items is very stable. A slightly increasing trend, especially for twenty clusters, can be identified when moving to sixty items. This implies that the computational performance worsens as the size of the warehouse grows. This trend becomes more evident when considering the instances containing one hundred items. However, it can also be seen that the warehouse configuration is not the driving factor affecting the running times, but rather the number of items and clusters. This also conforms to the design of the algorithm since the number of iterations performed is solely based on these two factors.

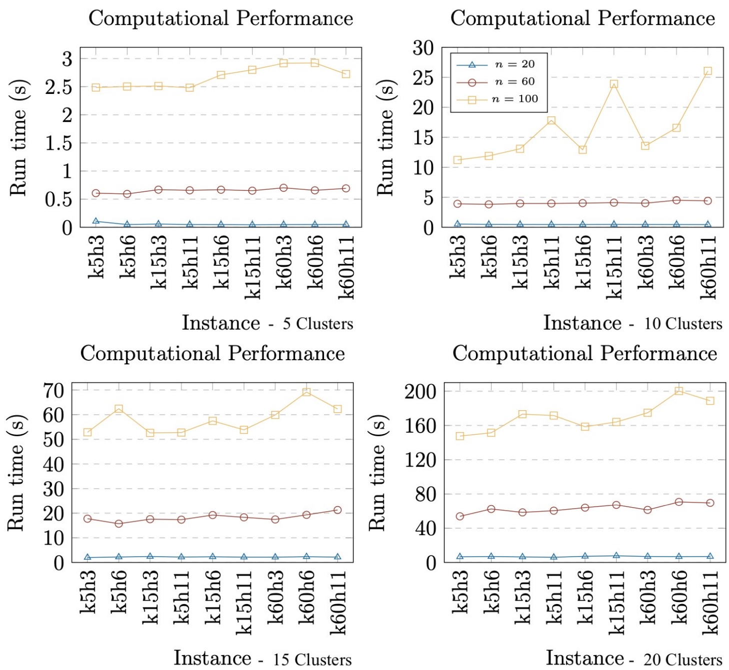


Figure 3.19: *Sensitivity analysis of Warehouse Layout*

Test Problem			MILP Model (Proposed) (Using CPLEX in GAMS)		MILP Model (Old) (Using CPLEX in GAMS)		MILP (Best Cost)		Constructive Greedy Search		Constructive Max-Min Search		VNS (Using best constructive)		Guided Local Search (GLS) +VNS			
Aisles	Cross Aisles	Products	Run Time* (s)	O.F. (Cost)	Run Time (s)	O.F. (Cost)	O.F. (Cost)	Optimality	Run Time (s)	O.F. (Cost)	Run Time (s)	O.F. (Cost)	Run Time (s)	O.F. (Cost)	Run Time (s)	O.F. (Cost)	Gap w.r.t. CPLEX (%)	Improve w.r.t. VNS (%)
5	**3**	**15**	4.96	2090	7.07	2090	2090	O	<1	2091	<1	2091	0.55	2090	0.15	2090	0,00	0,00
5	**3**	**60**	6.28	2549	7.53	2549	2549	O	<1	2554	<1	2552	1.48	2549	0.48	2549	0,00	0,00
5	**3**	**240**	1792.63	5066	2030.86	5066	5066	O	<1	5296	<1	5178	3.52	5074	0.72	5066	0,00	-0,16
5	**6**	**15**	10.57	2687	12.53	2687	2687	O	<1	2745	<1	2726	0.89	2687	0.39	2687	0,00	0,00
5	**6**	**60**	36.16	3400	46.16	3400	3400	O	<1	3681	<1	3741	2.61	3410	1.01	3400	0,00	-0,29
5	**6**	**240**	3600*	6796	3600	7042	6796	F	<1	5912	<1	5892	3.91	5580	1.34	5513	-18,88	-1,20
5	**11**	**15**	21.39	2757	27.23	2757	2757	O	<1	2759	<1	2757	1.77	2759	0.64	2757	0,07	0,00
5	**11**	**60**	40.73	4075	43.49	4075	4075	O	<1	4171	<1	4156	3.32	4078	1.11	4075	0,00	-0,07
5	**11**	**240**	3600	5333	3600	5585	5533	F	<1	5500	<1	5436	4.58	4328	1.20	4067	-23,74	-6,03
15	**3**	**15**	6.82	2465	7.47	2465	2465	O	<1	2471	<1	2477	2.09	2465	0.87	2465	0,00	0,00
15	**3**	**60**	50.79	3944	81.59	3944	3944	O	<1	4010	<1	3990	2.58	3944	0.98	3944	0,00	0,00
15	**3**	**240**	3600	7856	3600	8132	7856	F	<1	7584	<1	7492	3.97	6890	1.07	6598	-16,01	-4,24
15	**6**	**15**	16.79	3497	20.17	3497	3497	O	<1	3523	<1	3550	3.11	3497	1.14	3497	0,00	0,00
15	**6**	**60**	67.43	6315	84.19	6315	6315	O	<1	6345	<1	6345	4.80	6315	1.22	6315	0,00	0,00
15	**6**	**240**	3600	17561	3600	18120	17561	F	<1	18800	<1	18650	5.94	13580	2.36	13465	-23,32	-0,85
15	**11**	**15**	18.71	5684	26.83	5684	5684	O	<1	5708	<1	5699	2.35	5684	1.03	5684	0,00	0,00
15	**11**	**60**	80.11	9642	91.98	9642	9642	O	<1	9679	<1	9700	4.93	9642	2.15	9642	0,00	0,00
15	**11**	**240**	3600	18365	3600	18956	19365	F	<1	17329	<1	17023	6.31	15471	2.67	13674	-25,54	-11,62
60	**3**	**15**	4.96	2293	6.45	2293	2293	O	<1	2365	<1	2320	3.59	2293	1.16	2293	0,00	0,00
60	**3**	**60**	112.80	3650	132.23	3650	3650	O	<1	3676	<1	3687	4.17	3661	1.71	3650	0,00	-0,30
60	**3**	**240**	3600	13211	3600	13750	13211	F	<1	12432	<1	12560	5.08	11045	2.32	10621	-19,60	-3,84
60	**6**	**15**	43.18	3645	51.55	3645	3645	O	<1	3689	<1	3674	4.55	3645	1.86	3645	0,00	0,00
60	**6**	**60**	156.25	4024	180.77	4024	4024	O	<1	4140	<1	4156	5.56	4024	2.09	4024	0,00	0,00
60	**6**	**240**	3600	19660	3600	19792	19660	F	<1	17328	<1	17506	6.26	15590	2.37	15065	-23,37	-3,37
60	**11**	**15**	123.39	4950	148.40	4950	4950	O	<1	5031	<1	5015	4.01	4955	1.91	4950	0,00	-0,10
60	**11**	**60**	163.79	6705	198.28	6705	6705	O	<1	6983	<1	6950	6.15	6745	2.95	6705	0,04	-0,55
60	**11**	**240**	3600	21260	3600	21361	21260	F	<1	18806	<1	18651	9.57	16517	3.49	15871	-25.35	-3,91

* Maximum run time has been set 3600 seconds for CPLEX solver in GAMS, and Heuristics in MATLAB

Table 3.5: *Result for instances with cluster size=1*

Test Problem			MILP Model (Proposed)		MILP Model (Old)		CPLEX (Best Cost)		Constructive Greedy Search		Constructive Max-min Search		VNS (Using best constructive)		Guided Local Search (GLS) +VNS			
Aisles	Cross Aisles	Products	Run Time (s)	O.F. (Cost)	Run Time(s)	O.F. (Cost)	O.F. (Cost)	optimality	Run Time(s)	O.F. (Cost)	Run Time(s)	O.F. (Cost)	Run Time(s)	O.F. (Cost)	Run Time (s)	O.F. (Cost)	Gap w.r.t. CPLEX (%)	Improve w.r.t. VNS (%)
5	**3**	**15**	103.96	1437	156.45	1437	1437	**O**	<1	1706	<1	1685	1.55	1437	0.61	1437	0,00	0,00
5	**3**	**60**	214.73	2195	300.95	2195	2195	**O**	<1	2500	<1	2459	4.48	2195	0.78	2195	0,00	0,00
5	**3**	**240**	3600	4991	3600	5080	4991	**F**	<1	5063	<1	5111	8.48	4524	1.29	4487	-10,10	-0,82
5	**6**	**15**	190.09	1789	263.15	1789	1789	**O**	<1	1823	<1	1860	3.43	1789	1.38	1789	0,00	0,00
5	**6**	**60**	267.98	4078	366.43	4078	4078	**O**	<1	4300	<1	4311	5.61	4140	1.59	4078	0,00	-1,50
5	**6**	**240**	3600	13840	3600	14280	13840	**F**	<1	14158	<1	14601	10.91	12115	2.71	11784	-14,86	-2,73
5	**11**	**15**	266.98	2671	311.17	2671	2671	**O**	<1	2773	<1	2791	9.77	2680	2.86	2671	0,00	-0,11
5	**11**	**60**	365.74	6807	460.97	6807	6807	**O**	<1	7215	<1	7165	12.32	6855	3.03	6807	0,00	-0,19
5	**11**	**240**	3600	17656	3600	18320	17656	**F**	<1	15154	<1	15630	18.58	13254	3.17	12704	-28,05	-4,15
15	**3**	**15**	137.95	2060	175.05	2060	2060	**O**	<1	2412	<1	2374	10.09	2060	1.28	2060	0,00	0,00
15	**3**	**60**	3600	4331	3600	4411	4331	**F**	<1	4650	<1	4790	16.58	4019	1.99	3985	-7,99	-0,85
15	**3**	**240**	3600	6403	3600	6640	6403	**F**	<1	6415	<1	6320	14.47	5961	2.63	5874	-8,26	-1,46
15	**6**	**15**	646.39	3841	810.47	3841	3841	**O**	<1	4260	<1	4178	12.11	3932	2.07	3841	0,00	-0,72
15	**6**	**60**	3600	8412	3600	8650	8412	**F**	<1	7890	<1	7546	11.80	6180	2.74	6054	-28,03	-2,04
15	**6**	**240**	3600	9991	3600	10219	9991	**F**	<1	9927	<1	9711	18.94	8361	3.98	8361	-16,31	0,00
15	**11**	**15**	1321.07	7016	1570.87	7016	7016	**O**	<1	8605	<1	8410	7.35	7218	1.21	7121	0,07	-0,89
15	**11**	**60**	3600	13247	3600	13600	13247	**F**	<1	10865	<1	11149	20.93	10691	3.46	10021	-24,35	-6,27
15	**11**	**240**	3600	15265	3600	16073	15265	**F**	<1	14170	<1	13905	26.31	11594	3.95	11461	-24,92	-1,15
60	**3**	**15**	1826.10	3751	2100.46	3751	3751	**O**	<1	4032	<1	4096	11.59	3794	2.13	3761	0,09	-1,13
60	**3**	**60**	3600	5641	3600	5960	5641	**F**	<1	5311	<1	5390	17.17	5050	2.82	5050	-10,48	0,00
60	**3**	**240**	3600	9410	3600	9740	9410	**F**	<1	9140	<1	8834	23.08	8165	3.01	8031	-14,65	-1,64
60	**6**	**15**	2873.40	4584	3018.71	4584	4584	**O**	<1	5760	<1	5814	13.55	4881	1.66	4591	0,08	-2,25
60	**6**	**60**	3600	9174	3600	9460	9174	**F**	<1	7978	<1	7674	19.56	6984	2.96	6984	-23,87	0,00
60	**6**	**240**	3600	14970	3600	15640	14970	**F**	<1	13791	<1	14268	28.26	12417	4.72	11941	-20,23	-3,83
60	**11**	**15**	3600	7941	3600	8234	7941	**F**	<1	8315	<1	7842	22.01	7102	3.14	7102	-10,57	0,00
60	**11**	**60**	3600	17410	3600	18764	17410	**F**	<1	15607	<1	15251	31.15	14107	3.96	13943	-19,91	-1,16
60	**11**	**240**	3600	36450	3600	38410	36450	**F**	<1	33687	<1	32964	39.57	29723	4.91	28641	-21,42	-3,64

* Maximum run time has been set 3600 seconds for CPLEX solver in GAMS, and Heuristics in MATLAB

Table 3.6: *Result for instances with cluster size=2*

Test Problem			MILP Model (Proposed)		MILP Model (Old)		CPLEX (Best Cost)		Constructive Greedy Search		Constructive Max-min Search		VNS (Using best constructive)		Guided Local Search (GLS)+VNS			
Aisles	Cross Aisles	Products	Run Time (s)	O.F. (Cost)	Run Time(s)	O.F. (Cost)	O.F. (Cost)	optimality	Run Time(s)	O.F. (Cost)	Run Time(s)	O.F. (Cost)	Run Time(s)	O.F. (Cost)	Run Time (s)	O.F. (Cost)	Gap w.r.t. CPLEX (%)	Improve w.r.t. VNS (%)
5	3	15	589.54	1346	654.44	1346	1346	O	<1	1650	<1	1696	6.05	1346	0.74	1346	0,00	0,00
5	3	60	2497.54	2565	2658.54	2565	2565	O	<1	3320	<1	3396	7.08	3132	2.48	2565	0,00	-18,10
5	3	240	3600	5331	3600	5631	5331	F	<1	5931	<1	5976	25.08	5714	3.48	4832	-9,36	-15,44
5	6	15	1125.57	2065	1345.52	2065	2065	O	<1	2179	<1	2236	9.33	2065	1.43	2065	0,00	0,00
5	6	60	3241.65	2723	3380.54	2723	2723	O	<1	3871	<1	3954	13.67	3761	1.61	2723	0,00	-27,60
5	6	240	3600	NS	3600	NS	-	NS	<1	57946	<1	58388	29.81	54891	3.91	50328		-8,31
5	11	15	1948.54	3060	2248.54	3060	3060	O	<1	3410	<1	3320	13.98	3083	2.77	3083	0,75	0,00
5	11	60	3600	6551	3600	6743	6551	F	<1	6671	<1	6590	17.61	6504	3.32	6252	-4,56	-3,87
5	11	240	3600	NS	3600	NS	-	NS	<1	15743	<1	17679	34.64	15047	4.58	12134		-19,36
15	3	15	1195.43	1267	1305.54	1267	1267	O	<1	1532	<1	1696	10.63	1498	2.09	1287	1,58	-14,09
15	3	60	3600	3123	3600	3405	3123	F	<1	4106	<1	4210	13.33	3875	2.58	3087	-1,15	-20,34
15	3	240	3600	6733	3600	6974	6733	F	<1	12425	<1	13158	36.38	12425	3.47	6543	-2,82	-47,34
15	6	15	884.54	4125	1001.54	4125	4125	O	<1	4889	<1	4968	10.92	4889	2.11	4125	0,00	-15,63
15	6	60	3600	5691	3600	5793	5691	F	<1	5671	<1	5741	15.55	5664	1.80	5603	-1,55	-1,08
15	6	240	3600	NS	3600	NS	-	NS	<1	28919	<1	29828	40.93	25861	2.94	22851		-11,64
15	11	15	764.54	6239	805.54	6239	6239	O	<1	8015	<1	7371	29.56	7431	1.35	6365	2,02	-14,35
15	11	60	2554.54	11236	2740.54	11236	11236	O	<1	14658	<1	15432	32.21	12651	3.93	11242	0,05	-11,14
15	11	240	3600	14651	3600	15651	15651	F	<1	25715	<1	24375	43.39	20651	4.31	13542	-7,57	-34,42
60	3	15	1435.54	1895	1676.12	1895	1895	O	<1	2563	<1	2658	12.15	2321	2.59	1976	4,27	-14,86
60	3	60	3600	4563	3600	4695	4563	F	<1	4630	<1	4480	23.09	4400	3.17	3185	-30,20	-27,61
60	3	240	3600	NS	3600	NS	-	NS	<1	13476	<1	12876	45.63	11851	5.08	9954		-16,01
60	6	15	664.43	7827	984.21	7827	7827	O	<1	9958	<1	10719	15.70	9958	2.55	8051	2,86	-19,15
60	6	60	3120.54	10631	3600	10658	10631	O	<1	12868	<1	13314	21.92	12868	4.56	10631	0,00	-17,38
60	6	240	3600	NS	3600	NS	-	NS	<1	31891	<1	33149	51.12	30058	5.26	27412		-8,80
60	11	15	1841.43	16253	2254.54	16253	16253	O	<1	22281	<1	22734	25.00	19850	4.01	16464	1,30	-17,06
60	11	60	3600	NS	3600	NS	-	NS	<1	23461	<1	24187	35.98	22451	6.15	18540		-17,42
60	11	240	3600	NS	3600	NS	-	NS	<1	69460	<1	70481	59.53	56986	8.57	43541		-23,59

Table 3.7: *Result for instances with cluster size=5*

Test Problem			MILP Model (Proposed)		MILP Model (Old)		CPLEX (Best Cost)		Constructive Greedy Search		Constructive Max-min Search		VNS (Using best constructive)		Guided Local Search (GLS)+VNS			
Aisles	Cross Aisles	Products	Run Time (s)	O.F. (Cost)	Run Time(s)	O.F. (Cost)	O.F. (Cost)	optimality	Run Time(s)	O.F. (Cost)	Run Time(s)	O.F. (Cost)	Run Time(s)	O.F. (Cost)	Run Time (s)	O.F. (Cost)	Gap w.r.t. CPLEX (%)	Improve w.r.t. VNS (%)
5	**3**	**15**	1874.54	2310	2349.54	2310	2310	**O**	<1	3174	<1	2894	7.38	2509	1.22	2310	0,00	-7,93
5	**3**	**60**	3600	7517	3600	7936	7517	**F**	<1	7105	<1	7320	13.08	6578	1.98	6231	-17,11	-5,28
5	**3**	**240**	3600	NS	3600	NS	-	**NS**	<1	27565	<1	26173	24.49	23914	2.37	20431		-14,56
5	**6**	**15**	2334.65	4890	2895.31	4890	4890	**O**	<1	5730	<1	5978	13.35	5276	2.56	5004	2,33	-5,16
5	**6**	**60**	3600	10357	3600	10451	10357	**F**	<1	9972	<1	9650	18.52	8991	3.91	8934	-13,74	-0,63
5	**6**	**240**	3600	NS	3600	NS	-	**NS**	<1	35125	<1	36960	29.08	33671	6.32	30754		-8,66
5	**11**	**15**	3421.43	3841	3600	4007	3841	**O**	<1	5107	<1	4973	14.97	4621	3.72	3841	0,00	-16,88
5	**11**	**60**	3600	12749	3600	13651	12749	**F**	<1	11117	<1	10500	27.52	9244	5.09	8964	-29,69	-3,03
5	**11**	**240**	3600	NS	3600	NS	-	**NS**	<1	44734	<1	44926	39.11	40361	8.20	34651		-14,15
15	**3**	**15**	3490.54	3931	3567.42	3931	3931	**O**	<1	5502	<1	5438	11.77	4911	2.55	4259	8,34	-13,28
15	**3**	**60**	3600	10967	3600	11683	10967	**F**	<1	10863	<1	11950	16.01	9931	3.09	9143	-16,63	-7,93
15	**3**	**240**	3600	NS	3600	NS	-	**NS**	<1	36985	<1	35169	31.36	33596	4.11	27541		-18,02
15	**6**	**15**	3531.43	10583	3600	11127	10583	**O**	<1	13349	<1	15041	15.11	12460	3.65	11184	5,68	-10,24
15	**6**	**60**	3600	23194	3600	25076	23194	**F**	<1	20176	<1	19136	23.83	17490	6.99	15055	-35,09	-13,92
15	**6**	**240**	3600	NS	3600	NS	-	**NS**	<1	62324	<1	60645	36.14	56596	10.42	50654		-10,50
15	**11**	**15**	3600	17604	3600	18750	17604	**F**	<1	17312	<1	16681	20.78	13490	7.41	13490	-23,37	0,00
15	**11**	**60**	3600	39640	3600	41341	39640	**F**	<1	36431	<1	37173	32.84	33145	12.33	31884	-19,57	-3,80
15	**11**	**240**	3600	NS	3600	NS	-	**NS**	<1	77020	<1	74686	45.49	65199	18.17	60438		-7,30
60	**3**	**15**	3600	3701	3600	3970	3701	**F**	<1	3353	<1	4042	21.51	2318	4.30	2134	-42,34	-7,94
60	**3**	**60**	3600	NS	3600	NS	-	**NS**	<1	13480	<1	13100	36.87	11891	12.78	11650		-2,03
60	**3**	**240**	3600	NS	3600	NS	-	**NS**	<1	36914	<1	36187	45.35	33306	18.22	27651		-16,98
60	**6**	**15**	3600	9706	3600	10368	9706	**F**	<1	9271	<1	8045	24.28	7360	5.96	7164	-26,19	-2,66
60	**6**	**60**	3600	NS	3600	NS	-	**NS**	<1	22741	<1	21720	43.40	17831	13.58	16541		-7,23
60	**6**	**240**	3600	NS	3600	NS	-	**NS**	<1	48117	<1	49930	49.36	44247	21.37	40541		-8,38
60	**11**	**15**	3600	16580	3600	17803	16580	-	<1	17260	<1	15894	28.84	12760	10.31	12760	-23,04	0,00
60	**11**	**60**	3600	NS	3600	NS	-	**NS**	<1	35015	<1	33291	46.78	29990	16.43	27143		-9,49
60	**11**	**240**	3600	NS	3600	NS	-	**NS**	<1	88382	<1	85151	79.38	78365	29.08	71581		-8,66

Table 3.8: *Result for instances with cluster size=10*

Test Problem			MILP Model (Proposed)		MILP Model (Old)		CPLEX (Best Cost)		Constructive Greedy Search		Constructive Max-min Search		VNS (Using best constructive)		Guided Local Search (GLS)+VNS			
Aisles	Cross Aisles	Products	Run Time (s)	O.F. (Cost)	Run Time(s)	O.F. (Cost)	O.F. (Cost)	optimality	Run Time(s)	O.F. (Cost)	Run Time(s)	O.F. (Cost)	Run Time(s)	O.F. (Cost)	Run Time (s)	O.F. (Cost)	Gap w.r.t. CPLEX (%)	Improve w.r.t. VNS (%)
5	3	15	3600*	676	3600	783	676	F	<1	1013	<1	954	10.79	741	3.54	521	-22,93	-29,69
5	3	60	3600	4300	3600	4631	4300	F	<1	4023	<1	3917	16.19	3742	6.41	3370	-21,63	-9,94
5	3	240	3600	NS	3600	NS	-	NS	<1	13951	<1	13171	39.85	10637	10.34	7843		-26,27
5	6	15	3600	3634	3600	3810	3634	F	<1	3170	<1	3368	15.42	2657	5.20	2657	-26,88	0,00
5	6	60	3600	NS	3600	NS	-	NS	<1	24510	<1	26105	27.54	22695	10.43	18541		-18,30
5	6	240	3600	NS	3600	NS	-	NS	<1	69212	<1	72077	43.16	63971	15.54	59450		-7,07
5	11	15	3600	9684	3600	10848	9684	F	<1	8049	<1	7916	22.80	6049	7.54	5344	-44,82	-11,65
5	11	60	3600	NS	3600	NS	-	NS	<1	36971	<1	35200	35.61	32793	12.22	32793		0,00
5	11	240	3600	NS	3600	NS	-	NS	<1	66596	<1	65103	40.20	58250	24.65	52583		-9,73
15	3	15	3600	2341	3600	2694	2341	F	<1	2570	<1	2306	13.79	2022	4.25	2022	-13,63	0,00
15	3	60	3600	NS	3600	NS	-	NS	<1	14852	<1	16500	24.17	12851	7.54	10654		-17,10
15	3	240	3600	NS	3600	NS	-	NS	<1	59870	<1	57693	41.38	52176	18.32	47853		-8,29
15	6	15	3600	12960	3600	13912	12960	F	<1	12574	<1	12695	19.84	12167	9.31	10654	-17,79	-12,44
15	6	60	3600	NS	3600	NS	-	NS	<1	48054	<1	47220	36.58	43214	14.54	39544		-8,49
15	6	240	3600	NS	3600	NS	-	NS	<1	92914	<1	94045	18.37	88503	20.43	80541		-9,00
15	11	15	3600	17839	3600	18962	17839	F	<1	16420	<1	15970	26.41	14271	14.54	11563	-35,18	-18,98
15	11	60	3600	NS	3600	NS	-	NS	<1	65333	<1	67608	43.37	60419	17.59	60419		0,00
15	11	240	3600	NS	3600	NS	-	NS	<1	111978	<1	107248	79.96	100810	35.65	97546		-3,24
60	3	15	3600	3301	3600	3619	3301	F	<1	3029	<1	2980	16.68	2117	8.52	1947	-41,02	-8,03
60	3	60	3600	NS	3600	NS	-	NS	<1	14381	<1	15973	35.59	11170	19.65	10575		-5,33
60	3	240	3600	NS	3600	NS	-	NS	<1	126454	<1	120871	97.50	112691	35.43	107541		-4,57
60	6	15	3600	19730	3600	21035	19730	F	<1	19010	<1	19671	25.50	17493	21.54	16345	-17,16	-6,56
60	6	60	3600	NS	3600	NS	-	NS	<1	30087	<1	30643	48.12	26642	36.06	26642		0,00
60	6	240	3600	NS	3600	NS	-	NS	<1	144763	<1	142807	116.26	134196	44.65	121549		-9,42
60	11	15	3600	NS	3600	NS	-	NS	<1	36840	<1	39412	38.43	32540	23.48	32540		0,00
60	11	60	3600	NS	3600	NS	-	NS	<1	111539	<1	108941	72.78	99035	43.94	79553		-19,67
60	11	240	3600	NS	3600	NS	-	NS	<1	150767	<1	153491	130.92	142391	52.54	117584		-17,42

Table 3.9: *Result for instances with cluster size=20*

Test Problem			VNS		Guided Local Search	
Aisles	Cross Aisles	Products				
			Mean	Standard Deviation	Mean	Standard Deviation
5	**3**	15	541	44.94	521	17.49
5	**3**	60	3570	49.45	3570	402.30
5	**3**	240	10637	309.96	9843	376.64
5	**6**	15	2657	656.38	2657	201.65
5	**6**	60	22695	4730.74	20541	2153.96
5	**6**	240	63971	6085.55	62450	8345.54
5	**11**	15	6049	1724.37	5344	768.97
5	**11**	60	32793	338.88	32793	2691.73
5	**11**	240	58250	7667.06	56583	1176.57
15	**3**	15	2022	231.45	2022	45.28
15	**3**	60	12851	2951.30	10654	411.52
15	**3**	240	52176	12447.11	47853	6034.63
15	**6**	15	12167	682.10	10654	406.37
15	**6**	60	43214	6349.40	39544	4830.01
15	**6**	240	88503	11830.71	85541	3124.71
15	**11**	15	14271	2767.06	13563	1890.54
15	**11**	60	60419	12857.73	60419	3171.85
15	**11**	240	100810	22823.99	97546	2876.56
60	**3**	15	2117	175.30	2047	77.10
60	**3**	60	11170	2277.68	10575	977.20
60	**3**	240	112691	22147.09	107541	7634.69
60	**6**	15	17493	853.37	16345	862.18
60	**6**	60	26642	951.10	26642	3320.24
60	**6**	240	134196	20063.54	121549	10670.74
60	**11**	15	32540	9369.02	32540	2683.20
60	**11**	60	99035	10113.03	92553	12733.35
60	**11**	240	142391	25001.06	127584	5470.27

Table 3.10: *Mean & SD of GLS for large-scale instances*

3.6 Conclusion

In this chapter, we developed a new mathematical model in addition to a guided local search heuristic for the Generalized Travelling Salesman Problem (GTSP) with geographical overlap between clusters, with the application in order picking problem in the warehouse with scattered storage policy. This MILP model and the proposed GLS algorithm can be used for any warehouse layout (general graph), which makes this method more practical. To the best of our knowledge, for this problem, no exact algorithm for warehouses with more than two blocks exist in the literature. However, our proposed model can be implemented in warehouses having many more than two blocks. The motivation for this research stems not only from the potential benefits that can be achieved within the efficiency of a warehouse, but also because of the fact that very little attention has been given to this problem when accounting for scattered storage policy where the clusters overlap.

The proposed algorithm exploits problem-specific information during the search procedure to help guide the local search operators to promising areas of the search space. Based on the computational results, the algorithm performs very well. When considering a larger number of clusters, the algorithm finds the optimal solution in most cases, which is guaranteed by the exact methods. The algorithm often obtains a better solution over the best known in the remaining cases. It has been shown that the proposed GLS applies to a wide variety of warehouse configurations and can obtain high-quality solutions within seconds or at most a few minutes. Moreover, for solving the large scale instances with up to 4800 nodes, we implemented our GLS heuristic which is significantly fast (with computation time less than one minute). Moreover, the quality of the solutions for medium to large instances are much better than our MILP. Our algorithm is able to solve all the instances and gives very good solutions in a very short amount of time.

A significant limitation of this study is the implementation of the two exact formulations used to obtain the results for the instances, which are used to judge the quality of the proposed heuristic algorithm. The limitation stems directly from the implementation of the subtour elimination constraints. The number of subtour elimination constraints grows exponentially with the number of clusters. Hence the way in which these are implemented in the commercial solver significantly determines the computational performance. As a result, many instances containing fifteen and twenty clusters were not solved to optimality and retained a large GAP following the time limit of 3600 seconds. This not only restricted the analysis that could be made on the solution quality and computational performance between the heuristic and exact methods, but also on the size of the instances which could be considered. Furthermore, as a result of not being able to judge the exact quality of the solutions, the analysis which could be done on the performance across various warehouse layouts was also affected.

Future researches can consider due dates for order list, dynamic customer orders, zoning and multiple pickers joint problems. Development of problem-specific solution procedures, integration of batching/order selection decisions and inclusion of uncertain expected orders should also be taken into account in the decision-making process. Furthermore, future research could also focus on the pre-processing (graph reduction) in the presence of scattered storage policy where the clusters overlap.

4

The periodic multi-commodity service network design problem with regular and express deliveries under demand uncertainty

Adapted from: [. Rajabighamchi, van Hoesel, and Defryn [131]]

Abstract

This chapter studies the routing of multiple commodities (shipments) through a network with the aim to minimize the total cost. To transport these commodities from their origin to their destination hub, a combination of different services can be used, including scheduled trucks (following a dedicated trajectory, similar to bus routes) and express delivery. Each commodity starts its itinerary at its origin hub and needs to arrive at its destination hub before its deadline. The following cost factors are considered in the model: a fixed cost as well as a distance-based travel cost for the scheduled truck services, a cost for express delivery between each pair of hubs based on the size of the commodity, and the inventory holding cost at each hub.

We first define the problem as a mixed-integer linear program (MILP). To solve this MILP, we apply a branch-and-price algorithm that relies on column generation. In a second phase, we extend our model formulation to also deal with demand uncertainty (i.e., the size of each shipment varies) and present a two-stage, scenario-based stochastic model which we also solve using the branch-and-price algorithm. To generate the scenarios for the stochastic model, we apply Sample Average Approximation (SAA). Extensive computational experiments, including a sensitivity analysis are presented.

4.1 Introduction

In today's international context, the planning and coordination of all necessary logistics operations within a supply network is a tedious task. As supply chain partners often established strong dependencies on each other — with the aim to improve overall efficiency of the network —, any delay or disruption in the transport flows between these partners will create a significant impact on the underlying operations [4, 52, 37].

To plan and execute all required logistical operations within the supply chain, companies rely on third party logistics service providers (3PLs). These 3PLs manage the flow of goods between the different supply chain entities by either dispatching their own vehicles or by subcontracting logistics service providers to execute the required transportation requests [140, 118, 94].

The problem presented in this chapter is motivated by a case study in which a 3PL is responsible for coordinating all material flows that belong to the supply network of a large construction company within Europe (company names are confidential). The network consists of multiple hubs, which either take the form of transshipment points within the supply network or represent a local supply or demand node (potentially uniting multiple suppliers / customers within a certain region for simplicity). The flow density between each pair of hubs varies significantly over time (some connections are used only seldom, others have high volumes every day) and is uncertain (exact volumes are only known last-minute). As the 3PL does not have its own fleet of trucks, it relies on — often local — subcontractors (carriers) to execute the transports. It is worth mentioning that since the company did not provide us any real data or detailed information, at the end of this research, we were not able to show them any solution methods or compare the results with their case.

We distinguish two types of agreements between the 3PL and its subcontractors. First, there is a long-term agreement to establish a *periodic fixed capacity* on some of the network connections. For example, a truck is chartered every Monday and Thursday to drive a fixed trajectory. As these long-term commitments are valuable to the carriers (these provide a predictable income), competitive prices can be negotiated for the service. However, sufficient flow should be guaranteed over the link, as one should always pay for a full truckload, independent from the actual load. Second, the 3PL can book an *ad-hoc express delivery* on the spot market. This service is more flexible and its cost depends solely on the volume and trajectory of the actual load at a particular time (i.e., there is no long-term commitment here).

To model the decisions faced by the 3PL, we use a *service network design problem* (SNDP) formulation. SNDPs mainly support tactical decisions (e.g., fleet size, transport modes, ...) for the routing of commodities (such as goods, data, people, ...) within a network that consists of interconnected hubs and where the transport of a commodity occurs between its source and destination node. Variants of the SNDP have been successfully applied to many problems in, e.g., road transportation planning [105, 42, 139], railway planning [14, 102, 10, 178], flight scheduling [80, 19, 106] and telecommunication [116, 115, 111].

This chapter contributes to the academic literature in the following ways. First, we consider a service network design model over time to allow differentiation between the (periodic) scheduled truck services and the ad-hoc express delivery option. To the best of our knowledge, we are the first to distinguish these two transport modes with their individual cost structure. Second, we enrich the current state-of-the-art formulations by accounting for hub capacities and manage inventory levels accordingly. Third, we develop competitive solution approaches based on a branch-and-price algorithm with a column generation algorithm in each node to solve this realistic variant of the SNDP. Moreover, we extend our models and results to a setting with uncertain demand and present a two-stage, scenario-based stochastic model which is solved using the sample average approximation method. Finally, a broad range of managerial insights have been generated by means of an extensive sensitivity analysis.

The remainder of the chapter is organized as follows. In Section 4.2, the relevant literature is discussed. We present a formal problem statement and a mathematical model formulation in Section 4.3. Section 4.4 details a column generation solution approach for the deterministic problem variant. This model is extended towards stochastic demands in Section 4.5. The implementation of the models and an extensive set of computational experiments are presented in Section 4.6, after which we summarize the main conclusions and limitations of our research in Section 4.7.

4.2 Literature review

4.2.1 Service network design problem

In this section, we review the literature on the service network design problem (SNDP) as we identified this problem is most closely related to the main topic of this manuscript. Early research on the service network design problem dates back to Crainic and Rousseau [41] and Farvolden and Powell [55]. Since then, many researchers have been attracted to extend these models to incorporate more realistic problem features [142]. Since our focus is on exact solution approaches, we will limit ourselves to contributions from the literature in which such methods have been presented. For an overview of the state-on-the-art on heuristic and metaheuristic solution procedures, we refer the interested reader to Salimifard and Bigharaz [142].

Within the exact solution approaches, we distinguish two main research directions. First, there are methods that rely on *branching strategies*, such as branch-and-bound, branch-and-price(-and-cut) and column generation (see, e.g., Andersen et al. [6], Sarubbi et al. [145], Akyüz, Öncan, and Altınel [3], Boccia et al. [21] and Canel et al. [27]). Second, there are the contributions that focus on *decomposition-based methods* (see, e.g., Teypaz, Schrenk, and Cung [158], Oğuz, Bektaş, and Bennell [123], Rahmaniani et al. [130], Çakır [24] and Moradi, Raith, and Ehrgott [117]). In what follows, we highlight the most related and relevant contributions.

Boccia et al. [21] propose a *multi-commodity location routing problem* which they solve using a branch-and-cut algorithm. Given a set of potential facility locations and a set of demands (commodities), the multi-commodity location routing problem is about deciding how many and which of these facilities to open in order to minimize the total cost (i.e., a fixed cost for opening a facility and a variable cost based on the routing of the commodities) while covering all demand.

Wang et al. [175] propose a *service network design model* in which the routes for a heterogeneous fleet of vehicles should be determined, given a set of delivery points with predefined demand. The authors present both arc-based and path-based mathematical formulations to model the problem. To solve the problem, a hybrid algorithm is used that combines exact and heuristic techniques (including column generation, cutting planes and local search) to solve large-scale instances. The exact solvers within the algorithm are responsible for providing lower bounds and feasible solutions, whereas local search is used to generate feasible upper bounds. The presented computational experiments demonstrate the potential of a heterogeneous fleet in tactical planning, as this provides higher vehicle loading rates less unused capacity. In contrast to this study, the authors did not consider delivery times (deadlines), capacities in the hubs and demand uncertainty.

The *capacitated multi-commodity network design problem* is presented by Katayama [84]. In this problem variant, the arcs in the network have limited capacity. The decision maker also decides which arcs (and thus which of their corresponding capacities) to make available within the network. The total cost of the network — which is to be minimized — is given by the routing cost for shipping all commodities from their source to their destination and the activation of an arc. The authors present a path-based formulation augmented with strong inequalities. They use column generation in combination with an arc capacity scaling (i.e., a linear approximation on the use of arc capacity) and local branching (i.e., improve the quality of the relaxed model by also considering neighbouring solutions) to solve the problem. In this chapter, we do not restrict the arc capacity — even though individual vehicles do have capacities, we do not restrict the amount of vehicles that can travel on a certain arc. However, we do consider capacity restrictions in the hubs. As the opening of hubs is a long-term (strategic) decision, we do not consider the opening/closing of hubs nor flexibility in the available capacity.

Trivella et al. [163] develop a mathematical path based model formulation for the *multi-commodity network flow problem with soft transit times*. The model explicitly discourages the use of long commodity routes by means of a penalty for delays. The authors present a column generation approach to solve the problem. The economic implications on costs and delays for different definitions of the penalty functions are discussed within a context of the liner shipping industry. In Çakır [24], the authors use Benders decomposition to solve the *multi-commodity, multi-mode distribution planning problem*. In this multi-commodity flow problem, commodities do not have a dedicated source node but some nodes are labeled as general source node. Consequently, the demand of each destination node can be fulfilled from any source node.

4.2.2 Demand uncertainty and stochastic models

In this section, we review the most relevant research contributions concerning stochastic network design and network flow optimization models under uncertainty. Most stochastic models that focus on demand uncertainty make use of *two-stage stochastic programming* [38]. In a first stage — before the realization of the stochastic demand — these models (partially) set the values of some of the decision variables that need to be fixed before the true demand can be observed. Hence, these decisions are scenario-independent, yet they typically put constraints on the further (scenario-dependent) decisions. Thus the first stage variables are not directly influenced by the uncertainty while considering also the expected cost of the second stage model (i.e., after the realization of the stochastic demand). Two-stage stochastic programming was introduced by Dantzig [45] and has been applied successfully to tackle different supply chain problems [87, 12].

The *multi-commodity redistribution problem* with stochastic supply, demand and network is studied by Gao and Lee [58]. The authors focus on the redistribution of commodities to respond to different realizations of the demands. To solve the problem, the authors make use of a two-stage, scenario-based stochastic programming model. In the first stage, the authors minimize the total dissatisfaction cost (unmet demand and oversupply) over different demand and supply scenarios. In the second stage, the authors vary the network availability and minimize the total response time.

Barbarosoğlu and Arda [16] use a two-stage stochastic programming model to optimize the transport of first-aid commodities to disaster-affected areas. A multi-commodity, multi-modal network flow formulation is developed to describe the flow of material over an urban transportation network. The random variables in this study are dedicated to the resource requirements, which are assumed uncertain after a disaster has occurred. Hamdan and Diabat [69], then, apply a two-stage stochastic model to plan the production, inventory and location decisions in a red blood cell supply chain under demand uncertainty. Similarly, Dillon, Oliveira, and Abbasi [51] propose a two-stage stochastic model for inventory management in a blood supply chain by considering uncertain demand.

The generation of scenarios — as well as determining the optimal number of scenarios — largely affects the performance of stochastic programming models. These decision should therefore be taken with care. Löhndorf [103] review the most common methods for scenario generation in the context of stochastic programming, including the *quasi-Monte Carlo method, moment matching, methods based on probability metrics*, and the *Voronoi cell sampling method*. The Monte Carlo method — better known as *Sample Average Approximation (SAA)* — is a well-known approach to reduce the size of stochastic optimization problems by considering a subset of (preferably independent) scenarios after which a deterministic problem is solved for each of these. For a clear guide to SAA, we refer the interested reader to Kim, Pasupathy, and Henderson [85]. Within the context of supply chain network design, Santoso et al. [144] make use of the SAA scheme. In combination with an accelerated Benders decomposition algorithm, they can compute solutions to large-scale problems with a huge number of scenarios.

Sörensen and Sevaux [155] study a stochastic vehicle routing problem. The authors propose a method to combine a sampling-based approach to estimate the robustness or flexibility of a solution with a metaheuristic optimization technique, which allowed them to solve large problems with more complex stochastic structures. Mendoza et al. [112] propose a bi-objective multi-commodity vehicle routing problem with stochastic demand. The goal is to simultaneously minimize the total expected cost of a set of routes and the coefficient of variation. The authors use chance constraints to make sure that the probability of a route duration is less than its maximum given threshold. Monte Carlo simulation is applied for the feasibility check of these chance constraints.

Based on the presented literature review, we conclude that only few mathematical programming models for variants of the service network design problem have been proposed. Moreover, these models lack the inclusion of important real-life problem features such as heterogeneous vehicles (more specifically the inclusion of the possibility to use express delivery services), multi-commodity problems over time in which commodities have dedicated release times and deadlines, capacitated hubs and periodicity of the planning over time. The current manuscript aims to fill this research gap by considering the mentioned elements in the context of a multi-commodity service network design problem.

4.3 Single-period service network design problem

In this section, we formally present the *multi-commodity service network design problem with regular and express deliveries for a single period* (typically a week). This period is subdivided in multiple time intervals (e.g., days) for each of which a decision has to be made regarding the capacity on the individual network links and the flow over each link. For now, we will limit ourselves to a deterministic variant of the problem.

4.3.1 Mathematical notation and model assumptions

We are given a network, represented by the complete graph $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{A})$ in which $\boldsymbol{V}$ is the set of vertices (hubs) and $\boldsymbol{A}$ the set of arcs. For each $(i, j) \in \boldsymbol{A}$, c_{ij} and τ_{ij} represent the cost associated with traversing the arc and the travel time, respectively. Inside each hub, we distinguish two different processes: storage and cross-docking. Storage refers to the possibility to store shipments over multiple time intervals (i.e., the arrival time interval of the shipment is different from the departure time interval). For each vertex $i \in \boldsymbol{V}$, the storage capacity is limited and denoted by Q_i^V. Cross-docking refers to the process of receiving, sorting, recombining and dispatching incoming shipments within the same time interval, usually within a few hours [141]. As these activities do not make use of the internal storage space of the hub, we do not limit these by the hub capacity.

Let $\boldsymbol{K}$ be the set of all shipments (commodities) that should be served by the network. For each shipment $k \in \boldsymbol{K}$, O_k and D_k denote the source (origin) and destination node, respectively. The volume of the shipment is denoted by q_k. We allow the splitting of this volume such that partial customer orders can be transported via a different route through the network, if desirable. Furthermore, each shipment has a release time l_k, defined as the time at which the shipment becomes available at its source, and a dispatching time u_k, at which the shipment will be sent from its destination hub to the customer. This dispatching time can be interpreted as a hard deadline for the transport activities within the network related to this shipment. In case the shipment arrives at the destination hub before its dispatching time, it will be temporarily stored in inventory.

To execute the necessary transportation requests, the logistics service provider can choose between the following three transport options:

1. *Scheduled truck service*: based on long-term contracts, a dedicated capacity is available on certain routes in the network. Because of the long-term stability of these routes, competitive prices can be negotiated for installing the capacity. Let F denote the fleet of scheduled trucks, each with a capacity Q^F. For this service, we incur both a fixed cost for establishing a truck connection and a distance-based variable cost, denoted by c^F and c_{ij}, respectively. There should not be a one-to-one relationship between a shipment and a scheduled truck service as a shipment can switch to another truck at any hub.

2. *Express delivery*: The full transport of the shipment can be outsourced to a third-party logistics provider at a fixed rate based on the origin – destination as well as the volume of the shipment. This option is more expensive than using the capacity of the scheduled truck service, but offers more flexibility.

3. *Mixed scenario*: To execute the required transport operations, a combination of scheduled truck services and express delivery can be used. This means that

for certain connections the scheduled truck service will be used, whereas other parts of the itinerary will be covered using the express service.

The goal is to decide on the required capacity for the scheduled truck service and design the corresponding routes for these vehicles. Here, the decision maker trades off installing more capacity on the scheduled truck service versus accepting the (higher) costs of express delivery. Over high demand connections, the service providers likely prefer the scheduled truck service as loading rates can be high and the fixed cost of establishing the connection can be divided over a larger volume. For low demand connections, it might not be worth installing a scheduled truck service and an express delivery will then be preferred.

4.3.2 Mixed Integer Linear Programming formulation

We now model the deterministic network design problem with express deliveries as a mixed integer linear problem. Assuming that each period is identical, we will focus on a single period with T time intervals. For example, T can represent one week, which can be subdivided in 7 days, denoted by $t = \{1, \dots, 7\}$. By imposing that the status of the network (i.e., amount of truck available in each hub) at the end of the period equals the initial status, the logistics plan can easily be repeated for each consecutive period.

The following decisions have to be made:

1. The total *number of scheduled trucks* available at hub i at time t, denoted by f_{it}.
2. The *routes covered by the scheduled trucks*, represented by decision variable z_{ijt}, denoting the number of scheduled trucks traversing arc (i, j) at time t.
3. The *itinerary for each shipment* $k \in \boldsymbol{K}$, based on the following decision variables:
   - The quantity of shipment k shipped over arc (i, j) using a scheduled truck service at time t denoted by x^k_{ijt}.
   - The quantity of shipment k shipped over arc (i, j) using the express delivery service at time t, denoted by e^k_{ijt}.
4. *Inventory decisions within each hub*, given by the quantity of shipment k kept in inventory at hub i at time t, denoted by I_{ikt}.

To allow tractability of the model and avoid (unnecessary) complexity, the model is built according to the assumption that no partial shipments can be handed over to the next period. This means that all shipments must be handled within the period under consideration and — consequently —- that all shipments are assumed to have a release time and delivery date within the current period T. Within our model formulation, this also means that all inventory levels will equal zero at the start and the end

Table 4.1: *Mathematical notation for the MILP formulation of the deterministic multi-commodity network design problem with express deliveries.*

Sets	
$\boldsymbol{V}$	The set of all vertices (hubs) in the network.
$\boldsymbol{A}$	The set of all arc (i, j), with $i, j \in \boldsymbol{V}$.
$\boldsymbol{K}$	The set of all shipments that should be served by the network.
$\boldsymbol{T}$	The set of all time intervals.
Parameters	
Q_i^V	The storage capacity of hub i.
h_i	Inventory cost per time interval per unit of volume at hub i.
c_{ij}	The cost to traverse arc (i, j) with a scheduled truck.
τ_{ij}	The travel time over arc (i, j) for a scheduled truck.
τ_{ij}^E	The travel time for shipping by express over arc (i, j).
O_k	The source node (origin) for shipment k.
D_k	The destination node for shipment k.
q_k	The volume of shipment k.
l_k	The release time of shipment k.
u_k	The time at which the shipment will be dispatched from its destination hub to the customer.
c_{ij}^E	The cost per volume-unit to use express delivery on arc (i, j).
Q^F	Capacity of a scheduled truck.
c^F	Fixed cost for establishing a scheduled truck.
Decision variables	
f_i^0	The number of scheduled trucks available at hub i at the beginning of the time horizon.
f_{it}	The total number of scheduled trucks that remain at hub i at the end of time t.
z_{ijt}	The number of scheduled trucks traversing arc (i, j) at time t.
e_{ijt}^k	The volume of shipment k shipped by express mode over arc (i, j) at time t
x_{ijt}^k	The volume of shipment k shipped by scheduled truck over arc (i, j) at time t.
I_{ikt}	The volume of shipment k kept in inventory at hub i at time t.

of the period T. The assumption can be justified by the fact that the presented model has the purpose to support tactical (or even strategical) decisions with respect to the long-term contracts and required capacities for the scheduled truck services. In this respect, shipments can be generated (e.g., based on historical traffic data) such that they represent the partial trips typically covered within a single period. For operational decision support (e.g., the day-to-day dispatching of shipments), other methods can be used that take the established capacity of the scheduled truck service as given and optimize loading rates and costs based on, e.g., a rolling time window approach.

The full MILP formulation of the deterministic network design problem with express deliveries is given below. We summarize all notation in Table 4.1.

$$\min \left[\sum_{i \in \boldsymbol{V}} c^F f_i^0 + \sum_{t \in \boldsymbol{T}} \left(\sum_{(i,j) \in \boldsymbol{A}} c_{ij} z_{ijt} + \sum_{(i,j) \in \boldsymbol{A}} \sum_{k \in \boldsymbol{K}} c_{ij}^E e_{ijt}^k + \sum_{i \in \boldsymbol{V}} \sum_{k \in \boldsymbol{K}} h_i I_{ikt} \right) \right] \tag{4.1}$$

$s.t.$

$$\sum_{k \in \boldsymbol{K}} x_{ijt}^k \leq Q^F z_{ijt} \qquad \forall (i,j) \in \boldsymbol{A}; \forall t \in \boldsymbol{T} \tag{4.2}$$

$$f_{it} = f_{i(t-1)} + \sum_{(j,i) \in \boldsymbol{A} | t - \tau_{ji} \geq 1} z_{ji(t-\tau_{ji})} - \sum_{(i,j) \in \boldsymbol{A}} z_{ijt} \qquad \forall i \in \boldsymbol{V}; \forall t \in \boldsymbol{T} \backslash \{1\} \tag{4.3}$$

$$f_{i1} = f_i^0 - \sum_{(i,j) \in \boldsymbol{A}} z_{ij1} \qquad \forall i \in \boldsymbol{V} \tag{4.4}$$

$$f_i^0 = f_{iT} \qquad \forall i \in \boldsymbol{V} \tag{4.5}$$

$$I_{ikt} - I_{ik(t-1)} + \underbrace{\left(\sum_{(i,j) \in \boldsymbol{A}} x_{ijt}^k - \sum_{(j,i) \in \boldsymbol{A}} x_{ji(t-\tau_{ji})}^k \right)}_{scheduled\, truck} + \underbrace{\left(\sum_{(i,j) \in \boldsymbol{A}} e_{ijt}^k - \sum_{(j,i) \in \boldsymbol{A}} e_{ji(t-\tau_{ji}^E)}^k \right)}_{Express}$$

$$= \begin{cases} 0 & \forall k \in \boldsymbol{K}; \forall t \in [l_k, u_k]; \forall i \in \boldsymbol{V} \backslash \{O_k, D_k\} \\ q_k & \forall k \in \boldsymbol{K}; t = l_k; i = O_k \\ -q_k & \forall k \in \boldsymbol{K}; t = u_k; i = D_k \end{cases} \tag{4.6}$$

$$I_{ikt} = 0 \qquad \forall i \in \boldsymbol{V}; \forall k \in \boldsymbol{K}; \forall t \notin [l_k, u_k] \tag{4.7}$$

$$\sum_{k \in \boldsymbol{K}} I_{ikt} \leq Q_i^V \qquad \forall i \in \boldsymbol{V}; \forall t \in \boldsymbol{T} \tag{4.8}$$

$$e_{ijt}^k, x_{ijt}^k, I_{ikt} \geq 0 \qquad \forall i,j \in \boldsymbol{V}; \forall t \in \boldsymbol{T}; \forall k \in \boldsymbol{K} \tag{4.9}$$

$$f_i^0, f_{it}, z_{ijt} \in \mathbb{N} \qquad \forall i,j \in \boldsymbol{V}; \forall t \in \boldsymbol{T} \tag{4.10}$$

Objective function. The goal is to minimize the total cost of running the network over the full planning horizon T, given in Equation (4.1). This objective function contains the following terms: the sum of the fixed and variable cost related to the scheduled trucks, the cost of using express deliveries for the (partial) shipments that are not transported using the scheduled truck and the inventory holding costs at the hubs. Note that the total amount of scheduled trucks established in the network is given by the sum of all trucks initiated at the hubs at the start of the period, denoted by $\sum_{i \in \boldsymbol{V}} f_i^0$.

Constraints. To ensure feasibility of the network, the following constraints with respect to the truck routes, the itineraries of the commodities and the inventories in the hubs should be satisfied. Constraints (4.2) ensure that for each arc at each time the total flow dedicated to the scheduled truck service does not exceed the scheduled truck capacity available on the arc. Constraints (4.3) take care of the allocation of scheduled trucks over the different hubs in the network at each time interval. The number of trucks available at hub i at the end of time t is given by the amount trucks stationed at this hub at the end of $t-1$ plus the incoming trucks minus the outgoing trucks. The initial allocation of trucks at the start of the period is given by constraints (4.4). To allow the schedule to be repeated over time, the starting configuration is set equal to the ending configuration in constraints (4.5).

The inventory levels are controlled by constraints (4.6). These constraints define the (partial) amount of shipment k in different hubs (potential transshipment points, origin and destination) over time. Once the shipment has been released into the system ($t \geq l_k$), this amount equals the total volume of the shipment received in each hub minus what has left the hub either via a scheduled truck or express delivery. By means of constraints (4.7), we explicitly set all inventory levels to zero for times that the shipment is not active in the network (i.e., before its release time and after its dispatching time). Constraints (4.8) control the storage capacity of each hub.

Finally, the domain of the decision variables is set by constraints (4.9) and (4.10).

4.4 Branch-and-price algorithm

Branch-and-price (BP) algorithms embed dynamic column generation into a branch-and-bound framework to solve a MILP. We apply a *best-first branching strategy* on the number of trucks on each arc, denoted by z_{ijt}.

In each node of the search tree, we apply the column generation algorithm to solve the linear problem relaxation (relaxing the integrality constraint on the z_{ijt} variable). Each time no additional columns (routes) improve the master problem and the LP-relaxed solution does not satisfy the integrality conditions we use bounding on each branch and solve two separate column generation algorithms for each branch as follows: $z_{ijt} < \lfloor z_{ijt} \rfloor$ and $z_{ijt} \geq \lceil z_{ijt} \rceil$.

In what follows, we will go deeper on the column generation approach that is at the core of each node in our branch-and-price algorithm. The MILP model presented above aims to integrate the routing decisions for the scheduled trucks and — if desirable — the use of the express delivery service with the individual (partial) routes for each commodity flowing through the network. As a result, it easily becomes intractable, even for small instances.

To decouple the complexity of finding good routes for the vehicles from the routes of the commodities, we will rely on a *Dantzig-Wolfe decomposition* [179] and solve the problem using a column generation framework in which a master problem and sub-problem (the pricing problem) are solved in an iterative way.

To initialize the column generation procedure, we start with the situation in which no scheduled truck routes are established and solve the master problem. As a result, all shipments will be sent directly from source to destination via a dedicated express delivery. Even though this solution is feasible, it is likely not optimal as no bundling opportunities are seized, even not for shipments with the same origin and destination.

The sub-problem aims to find promising routes for each individual shipment for which a scheduled truck service can be used. By focusing solely on the most promising routes for an individual shipment, the size of the problem is kept as small as possible and many high-quality routes can be added to the master problem in each iteration. Once routes have been generated by the sub-problem, the master problem is run again with the aim to route all shipments through the available network in an optimal way (i.e., select a combination of routes, potentially complemented with one or multiple express connections, for each shipment). This procedure is iterated until no more routes (columns) with negative reduced cost can be found.

4.4.1 Master problem

The master problem determines the flow of all shipments through the network using a combination of scheduled trucks or express delivery — defined as routes. These routes, denoted by $\boldsymbol{R}$, are dedicated to specific shipments (i.e., the set of routes available for shipment k is denoted by $\boldsymbol{R}_k \subset \boldsymbol{R}$) and generated by the sub-problem.

Each route $r \in \boldsymbol{R}_k$ is characterized by a binary parameter w_{ij}^{rt}, denoting whether the route r runs over the link (i, j) at time t. The quantity of shipment k transported using this route r is given by x_r^k. We summarize the additional notation for the master problem in Table 4.3.

The master problem is defined mathematically as follows:

Table 4.3: *Additional mathematical notation for the master problem.*

Sets	
$\boldsymbol{R}$	The set of all routes in the master problem.
$\boldsymbol{R}_k \subset \boldsymbol{R}$	The set of all routes of scheduled trucks for shipment k.
$\boldsymbol{A}_r$	The set of all arcs in route r.
Parameters	
w_{ij}^{rt}	Binary parameter denoting whether route r runs over the link (i,j) at time t.
Decision variables	
x_r^k	The amount of shipment $k \in K$ shipped via route $r \in R_k$.

$$\min \left[\sum_{i \in \boldsymbol{V}} c^F f_i^0 + \sum_{t \in \boldsymbol{T}} \left(\sum_{(i,j) \in \boldsymbol{A}} c_{ij} z_{ijt} + \sum_{(i,j) \in \boldsymbol{A}} \sum_{k \in \boldsymbol{K}} c_{i,j}^E e_{ijt}^k + \sum_{i \in \boldsymbol{V}} \sum_{k \in \boldsymbol{K}} h_i I_{ikt} \right) \right] \tag{4.11}$$

$s.t.$

$$\sum_{k \in \boldsymbol{K}} \sum_{r \in \boldsymbol{R}_k} x_r^k w_{ij}^{rt} \leq Q^F z_{ijt} \qquad \forall (i,j) \in \boldsymbol{A}; \forall t \in \boldsymbol{T} \tag{4.12}$$

$$I_{ikt} - I_{ik(t-1)} + \sum_{r \in \boldsymbol{R}_k} x_r^k \left(\sum_{(i,j) \in \boldsymbol{A}_r} w_{ij}^{rt} - \sum_{(j,i) \in \boldsymbol{A}_r} w_{ji}^{r(t-\tau_{ji})} \right) + \left(\sum_{(i,j) \in \boldsymbol{A}} e_{ijt}^k - \sum_{(j,i) \in \boldsymbol{A}} e_{ji(t-\tau_{ji}^E)}^k \right)$$
$$= \begin{cases} 0 & \forall k \in \boldsymbol{K}; \forall t \in [l_k, u_k]; \forall i \in \boldsymbol{V} \backslash \{O_k, D_k\} \\ q_k & \forall k \in \boldsymbol{K}; t = l_k; i = O_k \\ -q_k & \forall k \in \boldsymbol{K}; t = u_k; i = D_k \end{cases} \tag{4.13}$$

$$x_r^k \geq 0 \qquad \forall k \in \boldsymbol{K}; \forall r \in \boldsymbol{R} \tag{4.14}$$

Constraints (4.3), (4.4), (4.5), (4.7), (4.8), (4.9) and (4.10).

Objective function. The objective function of the master problem is equal to the objective function of the global MILP formulation, presented in Section 4.3.2. The function minimizes the total cost of the network, including the fixed and variable cost of the scheduled trucks, the cost for all express deliveries and the inventory holding cost in the hubs.

Constraints. To comply with the route-based formulation required for connecting the master and its sub-problem, we slightly adapted some of the constraints from the global MILP formulation.

Constraints (4.12) set the required amount of scheduled trucks that drive over arc (i, j) at time t, given by z_{ijt}, based on the total flow over the routes that make use of this arc. Similar to our global MILP, we assume that each shipment can be split in a continuous way over different routes.

Constraints (4.13) are the flow balancing constraints in which we account for the inventory at the hubs. The third term of the equation accounts for changes in the inventory related to the shipment flowing through the available scheduled truck routes. The last (fourth) term on the left-hand side of the equation accounts for express deliveries of the shipment from the current hub to other hub(s) in the network.

Finally, constraints (4.14) set the domain for the newly added decision variable x_r^k.

4.4.2 Route generation sub-problem

The aim of the sub-problem is to generate additional routes that can be added to the set $\boldsymbol{R}$ and considered by the master problem. A route is defined as a path of one or multiple arcs in our network. To generate many promising routes fast, we run the sub-problem for each shipment separately.

Let z be the objective function of Master Problem (MP). Moreover, let μ_{ijt} be the dual variables corresponding to the capacity constraint (4.12) and γ_{ikt} the dual variables for the flow balancing constraints (4.13).

We also define y_{ijt} as a binary variable that takes the value 1 if arc (i, j) is used at time t in the generated route, 0 otherwise. For modelling purposes, we also introduce S_{it} and E_{it} to represent the starting and ending node of the route, respectively. As the final itinerary of a shipment in the master problem can be defined as a combination of routes — potentially also including one or multiple express arcs —, we do not impose that the starting (or ending) hub of the generated routes coincide with the source (or destination) hub of the shipment.

Finally, let I_{it}^B be an auxiliary binary variable that takes the value 1 if there is a positive inventory in hub i at time t, 0 otherwise. Again, we summarize the additional mathematical notation in Table 4.5.

The *total reduced cost* for shipment k is computed using equation (4.15), which accounts for all reduced costs for the master problem constraints with the non-basic variable x_r^k, i.e., constraints (4.12) and (4.13).

Table 4.5: Additional notation for the sub-problem.

Parameters	
μ_{ijt}	Dual variable for constraints (4.12) of the master problem (capacity constraint).
γ_{ikt}	Dual variable for constraints (4.13) (inventory and flow balancing constraint).
Decision variables	
y_{ijt}	Binary variable that equals 1 if arc (i,j) is traversed at time t, 0 otherwise.
I^B_{it}	Binary variable that equals 1 if there is a positive inventory in hub i at time t ($\sum_{k \in \boldsymbol{K}} I_{ikt} > 0$), 0 otherwise.
S_{it}	Binary variable that equals 1 if the current route starts in hub i at time t, 0 otherwise.
E_{it}	Binary variable that equals 1 if the current route ends in hub i at time t, 0 otherwise.

$$Z^k = \sum_{t \in \boldsymbol{T}} \left(- \sum_{(i,j) \in \boldsymbol{A}} y_{ijt}\mu_{ijt} - \sum_{(i,j) \in \boldsymbol{A}} y_{ijt}\gamma_{ikt} + \sum_{(j,i) \in \boldsymbol{A}} y_{ji(t-\tau_{ji})}\gamma_{ikt} \right) \tag{4.15}$$

Then, the route generation sub-model is given by the following mathematical program.

$$\min \left[Z^k \right] \tag{4.16}$$

$s.t.$

$$\sum_{i \in \boldsymbol{V}} \sum_{t=(l_k+\tau^E_{O_k i})}^{(u_k-\tau^E_{iD_k})} S_{it} = 1 \tag{4.17}$$

$$\sum_{i \in \boldsymbol{V}} \sum_{t=(l_k+\tau^E_{O_k i})}^{(u_k-\tau^E_{iD_k})} E_{it} = 1 \tag{4.18}$$

$$S_{it} + E_{it} \leq 1 \qquad \forall i \in \boldsymbol{V}; \forall t \in \boldsymbol{T} \tag{4.19}$$

$$I^B_{it} - I^B_{i(t-1)} - S_{it} + E_{it} = \sum_{(j,i) \in \boldsymbol{A}} y_{ji(t-\tau_{ji})} - \sum_{(i,j) \in \boldsymbol{A}} y_{ijt} \qquad \forall i \in \boldsymbol{V}; \forall t \in \boldsymbol{T} \tag{4.20}$$

$$y_{ijt}, I^B_{it}, S_{it}, E_{it} \in \{0,1\} \qquad \forall i,j \in \boldsymbol{V}; \forall t \in \boldsymbol{T} \tag{4.21}$$

Objective function. The objective function of the sub-problem is the minimization of the reduced cost. If this optimal reduced cost is negative, the corresponding route will be added to the set of routes considered by the master problem.

Constraints. Constraints (4.17) and (4.18) ensure that routes have exactly one starting and one ending node, which are visited within a feasible time window for the shipment under consideration. Constraints (4.19) force the starting hub and time to be different from the ending hub and time.

The generated route should not only represent a path from start to end node, also the time at which different links are used should be consistent by considering intermediate storage in a hub if necessary, as seen in constraints (4.20).

Finally, constraints (4.21) take care of the domain constraints for the decision variables.

Once the sub-problem is solved for all shipments, the generated routes will be added to the master problem. This is done via the parameters w_{ij}^{rt}, which represent the y_{ijt} variables for each route.

4.5 Multi-period service network design problem

In the previous sections, we determined scheduled truck routes for a single period (e.g., week) with multiple time intervals. As these routes are established through long-term collaboration with dedicated carriers, they will be repeated every period. For example, if a scheduled truck connection is installed between two hubs during the first time interval of the period (e.g., Monday), this service will be provided every Monday. In this Section, we therefore extend the problem definition to a multi-period time horizon.

As demand is not constant, but might differ between periods, we need to establish the scheduled truck routes such that the overall long-term cost is minimized. If for most Mondays, e.g., the demand for connection A-B is rather low, we rather not install a scheduled truck over this connection on Monday (as this leads to large unused capacities during most weeks) but cover the demand with occasional express deliveries. If, on the other hand, demand is consistently high on Mondays, it might be beneficial to install a (less expensive) scheduled truck service that covers this connection.

To determine the optimal configuration of the scheduled truck services over multiple periods, we make use of a stochastic optimization model to account for the variability (and thus uncertainty) in the demand over time. More specifically, we employ a *two-stage scenario-based stochastic programming model* for this purpose.

4.5.1 Two-stage scenario-based stochastic programming model

The main idea behind the two-stage stochastic programming model is that we separate decision variables that depend directly on the scenarios (i.e., what will remain constant over the different periods) and are defined prior to the realization of the uncertain parameters, which occurs later in the second stage, from the decision variables that are impacted directly by the realization of the demand (i.e., what will change every period). The first set of variables are related to the scheduled truck routes, as these are part of a long-term collaboration and thus cannot be altered every period. The second set of variables relates to the volumes transported via the express delivery service as well as the inventories at the different hubs, as these will vary every period depending on the demand scenario.

The formulation of the two-stage problem assumes that the uncertain data (i.e., the demand realization) can be modelled as a random vector with a known probability distribution which remains constant over time. Consequently, one may reliably estimate the underlying probability distribution after which the optimization on the expected value could be justified by the law of large numbers [172, 60, 162]. Therefore, the objective function is to minimize the the first-stage costs while in addition to the expected value of the random second-stage costs.

We will model the demand realization by means of a random vector with a finite number of realizations (the scenarios). In our problem, the random vector consists of only one random parameter, being the demand (commodity size) between origin-destination pairs. Let $\mathbf{\Omega}$ be the set of scenarios (indexed by s). The probability of each scenario is denoted by $P(s)$, $\forall s \in \mathbf{\Omega}$. Additionally, we extend the decision variables denoting the flow in the network with an index s to account for the differences in demand for each shipment k between each scenario. For an overview of the additional notation, we refer to Table 4.7.

We extend our original mathematical problem to a two-stage stochastic programming model as follows.

$$\min \left[\underbrace{\left(\sum_{i \in \boldsymbol{V}} c^F f_i^0 + \sum_{(i,j) \in \boldsymbol{A}} \sum_{t \in \boldsymbol{T}} c_{ij} z_{ijt} \right)}_{\text{First stage}} + E\left[\varphi\right] \right] \tag{4.22}$$

$s.t.$

$$\sum_{k \in \boldsymbol{K}} x_{ijts}^k \leq Q^F z_{ijt} \qquad \forall (i,j) \in \boldsymbol{A}; \forall t \in \boldsymbol{T}; \forall s \in \mathbf{\Omega} \tag{4.23}$$

__Table 4.7:__ Additional notation for the two-stage scenario-based stochastic programming model.

Sets	
$\boldsymbol{\Omega}$	The set of all scenarios, indexed by s.
Parameters	
q_{ks}	The total volume of shipment k under scenario s.
Decision variables	
e^k_{ijts}	The (partial) volume of shipment k shipped by express mode over arc (i,j) at time t under scenario s.
x^k_{ijts}	The (partial) volume of shipment k shipped over arc (i,j) at time t under scenario s.
I^k_{its}	The (partial) volume of shipment k kept in inventory at hub i at time t under scenario s.

$$I^k_{its} - I^k_{i(t-1)s} + \left(\sum_{(i,j)\in \boldsymbol{A}} x^k_{ijts} - \sum_{(j,i)\in \boldsymbol{A}} x^k_{ji(t-\tau_{ji})s} \right) + \left(\sum_{(i,j)\in \boldsymbol{A}} e^k_{ijts} - \sum_{(j,i)\in \boldsymbol{A}} e^k_{ji(t-\tau^E_{ji})s} \right)$$
$$= \begin{cases} 0 & \forall k \in \boldsymbol{K}; \forall t \in [l_k, u_k]; \forall i \in \boldsymbol{V}\backslash\{O_k, D_k\}; \forall s \in \boldsymbol{\Omega} \\ q_{ks} & \forall k \in \boldsymbol{K}; t = l_k; i = O_k; \forall s \in \boldsymbol{\Omega} \\ -q_{ks} & \forall k \in \boldsymbol{K}; t = u_k; i = D_k; \forall s \in \boldsymbol{\Omega} \end{cases} \tag{4.24}$$

$$I^k_{its} = 0 \qquad \forall i \in \boldsymbol{V}; \forall k \in \boldsymbol{K}; \forall t \notin [l_k, u_k]; \forall s \in \boldsymbol{\Omega} \tag{4.25}$$

$$\sum_{k\in \boldsymbol{K}} I^k_{its} \leq Q^V_i \qquad \forall i \in \boldsymbol{V}; \forall t \in \boldsymbol{T}; \forall s \in \boldsymbol{\Omega} \tag{4.26}$$

$$e^k_{ijts}, x^k_{ijts}, I^k_{its} \geq 0 \qquad \forall i,j \in \boldsymbol{V}; \forall t \in \boldsymbol{T}; \forall k \in \boldsymbol{K}; \forall s \in \boldsymbol{\Omega} \tag{4.27}$$

Constraints (4.3), (4.4), (4.5), and (4.10).

Objective function. Equation (4.22) minimizes the cost associated with the first stage variables (i.e., the scheduled truck service) plus the expected value for the second stage cost. The latter cost is defined as a weighted sum of the cost of the outcome for each scenario multiplied by its respective probability, such that

$$E\left[\varphi\right] = \sum_{s\in \boldsymbol{\Omega}} P(s)\varphi_s. \tag{4.28}$$

The second stage cost of scenario s, denoted by φ_s, is given by the following equation. The first term accounts for the cost of all required express deliveries for the shipments that could not be transported entirely by the scheduled truck services. The second term relates to the inventory holding costs in each hub.

$$\varphi_s = \sum_{t \in \boldsymbol{T}} \left(\sum_{(i,j) \in \boldsymbol{A}} c^E_{ij} e^k_{ijts} + \sum_{i \in \boldsymbol{V}} \sum_{k \in \boldsymbol{K}} h_i I^k_{its} \right) \tag{4.29}$$

Constraints. Some constraints of the original model are updated to account for the different demand scenarios. Equation (4.23) ensures that under no scenario, the capacity of the arcs (with respect to the installed scheduled truck capacity) is violated. Constraints (4.24) connect the flows over the network links — both using scheduled trucks as well as express delivery — with the inventory at the different hubs over time. The inventory is initialized to zero for all time intervals a shipment is not in the system by constraints (4.25). Constraints (4.26) ensure the capacity of the hub is never exceeded. Finally, the domain constraints for the newly added decision variables are denoted by constraints (4.27).

4.5.2 Scenario generation using Sample Average Approximation

As the size of the network, the amount of shipments and the number of time intervals per period increase, the number of required scenarios to realistically represent the possible demand outcomes grows fast. To keep the model tractable, we make use of *Sample Average Approximation* (SAA). This technique relies on the generation of scenarios by means of Monte Carlo simulation [171]. Assuming that each scenario occurs with the same probability, we can rewrite equation (4.28) to

$$E\left[\varphi\right] = \frac{1}{|\Omega|} \sum_{s \in \boldsymbol{\Omega}} \varphi_s. \tag{4.30}$$

To construct our scenarios, we draw the volume of each shipment k, denoted by q_{ks}, from a normal probability distribution $N(\mu_k, \sigma_k)$.

Following Verweij et al. [171], Bagaram and Tóth [13], and Ahmed, Shapiro, and Shapiro [2], the steps that are considered within our SAA implementation are summarized in Table 4.9.

To test the relationship between the number of scenarios in the two-stochastic optimization model and the optimality gap, we conducted a computational experiment which is discussed in detail in Section 4.6.2.

Table 4.9: Different steps of the SAA implementation.

Step 1.	Initialize the number of independent samples, (SAA replications), denoted by M, as well as the sample size (number of scenarios)n. In other words, we generate M independent problems with n scenarios each, and then solve these M problems and calculate their average objective value. Each sample contains one value for the demand size between every origin-destination pairs. For defining the value of the M, we used the method presented in Ahmed, Shapiro, and Shapiro [2] which is based on the probability of the best improvement in the objective value. The sample size dictates the number of scenarios you will consider. The larger this value, the better the accuracy (lower optimality gap) at the expense of larger computing times. n is set to a small number such that $n << N$, in which N is defined as the largest sample size for which the stochastic model is tractable.
Step 2.	Generate M independent samples — each of size n — and solve the two stage stochastic problem for each sample.
Step 3.	Compute the mean and the variance of the results obtained in step 2. The average objective value is used as a lower bound for the stochastic problem.
Step 4.	Solve the stochastic model with N scenarios to find a (close to) optimal solution $\hat{x}$. Use this solution to set the first stage variables of the two-stage stochastic model for the M independent samples. Solve these models and again take the average objective function value now as an upper bound for the stochastic problem.
Step 5.	Compare the lower and upper bound computed in steps 3 and 4, respectively by determining the optimality gap which is the difference between upper and lower bound.
Step 6.	If the optimality gap found in step 5 is small enough, you stop. Otherwise, increase the sample size n and return to step 2.

4.6 Model implementation and computational experiments

4.6.1 Test instances

The algorithms presented in this chapter are tested extensively on a variant of the *Canad problem instances*[1] for the multi-commodity network design problem [43, 70]. In total, 41 instances are considered (from which 14 R and 27 C instances), which we altered to comply with our problem definition.

More specifically, we added a release and dispatching time (deadline) for each shipment as follows: first, we scale the time horizon to ensure that no dispatching time (deadlines) falls beyond the length of the period T. Then, for each shipment, we randomly select a release time such that the time difference between the release time and deadline is at least the transit time given in Hellsten et al. [70].

Furthermore, we also changed the cost structure of the instances to match the difference between the scheduled truck service and the express delivery option in the following way: the express cost per volume unit per arc is set in such a way that if the shipped volume is less than half of the truckload, then transporting the commodity by express mode is cheaper than shipping it by scheduled truck and vice versa. The scheduled truck capacity is fixed to 30 tons (based on long-haul transportation trucks). In Hellsten et al. [70], the fixed costs are given for each arc. To transform these into a fixed cost for a vehicle, we compute the shortest path for each shipment and take the average of the corresponding fixed costs of the respective arcs. For the variable cost, we take the unit flow costs given in the benchmark instances. Since this ratio is instance dependant (it varies when the network density, size and the distances changes), we can not set a unique value of variable cost and express cost for all the instances. Therefore, for each instance size, we compute the average variable cost for all the arc in the network ($\bar{c}_{ij} = \sum_{(i,j)\in A} \frac{c_{ij}}{|A|}$), and then depending on the instance size we select the value for express cost in such a way that the following equality holds:

$$c^E_{ij} = (c^F + \bar{c}_{ij})/(0.5Q^F * \bar{c}_{ij}) * c_{ij}$$

We refer to the fraction $\left(2\frac{c^F+\bar{c}_{ij}}{Q^F\bar{c}_{ij}}\right)$ as the express coefficient. As a result of this procedure, also the express costs are instance-dependent.

In all instances, a single period (week) contains 7 consecutive time intervals (days), so $t = \{1, 2, \ldots, 7\}$. The hub capacity is assumed to be 1000, and inventory cost per unit per day is 2.

[1]The original instances can be downloaded via `https://commalab.di.unipi.it/datasets/mmcf/#Canad` or `https://zenodo.org/record/4050442`.

All instances are solved using CPLEX 12.8 with default parameters on a MACBOOK AIR with an APPLE SILICON M1 CHIP and 16GB of RAM. Computational time is limited to 7200 seconds (2 hours).

4.6.2 Impact of the number of scenarios on the optimality gap

To test the impact of the number of scenarios on the optimality gap, we conducted a small computational experiment. We set M (the number of independent replications) equal to 30 and, given our hardware setup, N was found to be around 60 scenarios. We then varied the sample size n from 5 to 60. The results are summarized in Figures 4.1 and 4.2.

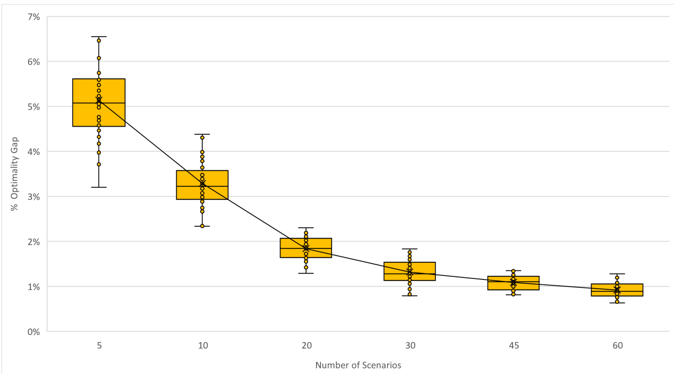


Figure 4.1: *Gap percentage for different number of scenarios n (with $N = 60$ and $M = 30$).*

Figure 4.1 shows the average optimal gap value for all the instances for different numbers of scenarios. The figure shows that as the number of scenarios increases, the solutions converge toward the optimal value, meaning that larger number of scenarios result in a lower optimality gap. However, as shown in Figure 4.2, larger sample sizes lead to a significant increase in solution time. The value on the y-axis represents the average solution time for all the instances (stochastic variant), divided by the objective function of the deterministic model (with only one scenario).

Based on this experiment, we conclude for our experiments that solving with up to 30 scenarios is sufficient to obtain close-to-optimal solutions (1.5% gap on average).

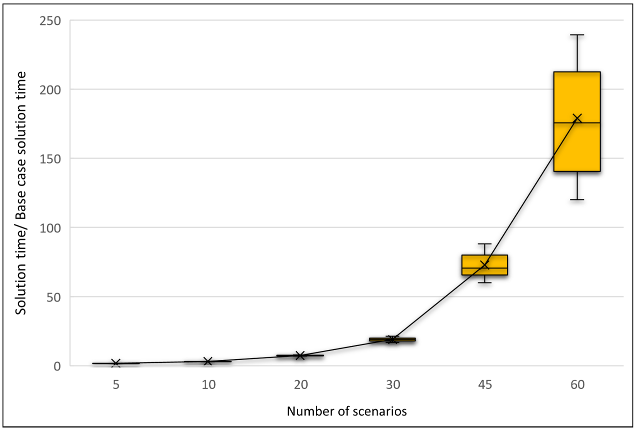


Figure 4.2: *(Relative) solution time vs number of scenarios*

4.6.3 Results for the deterministic single-period problem variant

Performance analysis of the branch-and-price algorithm

We analyse the performance of the branch-and-price algorithm on the different instances. A detailed overview of the results is presented in Tables 4.11 and 4.12.

Each instance is characterized by the number of hubs (nodes in the network), the number of arcs, and the number of shipments, denoted by $|\boldsymbol{V}|$, $|\boldsymbol{A}|$ and $|\boldsymbol{K}|$, respectively.

Next to the objective function value, we report on the number of columns added to the model (**#col**), the number of nodes in the branch-and-price tree (**#nodes**), the optimality gap (**Gap(%)**) and the computation time (**Time (s)**). The optimality gap is computed as follows:

$$\text{Optimality Gap (\%)} = \frac{\text{OptValue} - \text{Lower bound}}{\text{OptValue}} * 100$$

Based on the results in Tables 4.11 and 4.12, we see that the smaller instances (with ≤ 20 hubs and ≤ 50 shipments) are all solved to optimality within the two-hour time limit. Except for instance C36, a 3% optimality gap remains.

For the larger R instances (see Table 4.11) with more than 50 shipments, 1 instance (out of 6) is still solved to optimality (R11.1). An average optimality gap of 2.47% and 3.57% is found for the R instances with 100 and 200 shipments, respectively. With

Table 4.11: *Results for the branch-and-price algorithm on the deterministic single-period problem variant (R instances).*

Inst.	$\lvert \boldsymbol{V} \rvert$	$\lvert \boldsymbol{A} \rvert$	$\lvert \boldsymbol{K} \rvert$	**Obj.**	**#col**	**#nodes**	**Gap (%)**	**Time (s)**
R04.1	10	60	10	10200.95	148	216	0%	182.3
R07.1	10	82	10	10402.83	387	414	0%	489.4
R05.1	10	60	25	24442.39	500	392	0%	448.6
R08.1	10	83	25	22084.3	671	531	0%	232
R09.1	10	83	50	37401.92	498	706	0%	1318.2
						# optimal	5/5	
						Average	0%	534.1
R10.1	20	120	40	32728.53	1100	1822	0%	2193.6
R13.1	20	220	40	31497.12	2672	4263	0%	4058.5
R16.1	20	314	40	32549.78	3049	2544	0%	6244.7
						# optimal	3/3	
						Average	0%	4165.6
R11.1	20	120	100	81768.7	1290	3074	0%	6916.9
R14.1	20	220	100	74208.35	1033	935	2.6%	7200
R17.1	20	318	100	74266.11	2841	5029	4.8%	7200
						# optimal	1/3	
						Average	2.47%	7105.6
R12.1	20	120	200	152209.11	1528	1118	1.7%	7200
R15.1	20	220	200	133881.56	2924	3812	5%	7200
R18.1	20	315	200	132956.13	2002	3684	4%	7200
						# optimal	0/3	
						Average	3.57%	7200
						# optimal	**9/14**	
						Average	**1.29%**	**4148.87**

Table 4.12: *Results for the branch-and-price algorithm on the deterministic single-period problem variant (C instances).*

Inst.	$\|V\|$	$\|A\|$	$\|K\|$	**Obj.**	**#col**	**#nodes**	**Gap (%)**	**Time (s)**
C33	20	228	40	148638.88	2343	3574	0%	3752
C35	20	230	40	113698.45	1222	926	0%	4020.7
C36	20	230	40	139436.73	1217	1603	3%	7200
C41	20	288	40	137455.48	3112	5161	0%	3085.8
C42	20	294	40	161605.96	2533	2117	0%	3102.9
C43	20	294	40	137794.95	3205	5008	0%	3371.4
C44	20	294	40	153048.34	1844	1962	0%	3241.4
						# optimal	6/7	
						Average	0.43%	3967.74
C37	20	228	200	12172.44	1170	2409	5%	7200
C38	20	230	200	14931.77	1966	3277	0%	4422.6
C39	20	229	200	14436.66	1233	1493	6.9%	7200
C40	20	228	200	12703.92	1890	2855	4.5%	7200
C45	20	294	200	13589.56	2736	2719	10%	7200
C46	20	292	200	14197.69	2349	3901	13%	7200
C47	20	291	200	14376.92	1721	1355	4.4%	7200
C48	20	291	200	25285.64	2439	3984	108%	7200
						# optimal	1/8	
						Average	18.98%	6852.83
C49	30	518	100	14318.39	6928	13722	9.5%	7200
C50	30	516	100	13897.58	5821	7407	11%	7200
C51	30	519	100	14973.29	3829	3365	7.1%	7200
C52	30	517	100	16721.7	3271	7691	3.8%	7200
C57	30	680	100	13810.24	7419	21502	5.4%	7200
C58	30	680	100	13771.14	5516	7921	5.2%	7200
C59	30	687	100	26641.45	4528	11829	129%	7200
C60	30	686	100	13855.5	7709	22640	5.9%	7200
						# optimal	0/8	
						Average	22.11%	7200
C53	30	520	400	63779.25	4080	10428	169.2%	7200
C54	30	520	400	66407.1	2592	6116	184.1%	7200
C55	30	516	400	67828.75	6982	15208	187.5%	7200
C56	30	518	400	65287.05	4090	10382	178.4%	7200
						# optimal	0/4	
						Average	179.8%	7200
						# optimal	**7/27**	
						Average	**38.92%**	**6259.14**

an overall average optimality gap of 1.29%, we obtain competitive results for the R instances.

For the larger C instances (see Table 4.12) we see that the branch-and-price model is viable for most instances with 20–30 hubs and 100–200 shipments with optimality gaps below or around 10%. The average results are impacted heavily by the high optimality gaps for instance C48 (108%) and C59 (129%). The two-hour time limit is clearly insufficient to solve instances with $\geq$ 30 hubs or $\geq$ 200 shipments to optimality.

For the largest instances with 30 hubs and 400 shipments, we even notice very large optimality gaps. From this point onwards, the branch-and-price algorithm becomes really intractable.

The numerical results show promising outcomes in terms of convergence, speed and solution accuracy. Furthermore, the sensitivity analysis performed on key parameters demonstrated the robustness of the numerical model and its ability to capture subtle variations in the system. Based on the obtained results, we are satisfied with the performance demonstrated by our proposed mathematical model and the solution method. These results bolster our confidence in the chosen numerical approach and its suitability for investigating complex transport systems.

Impact of express delivery option

A unique feature of our problem formulation is the possibility to make use of an express delivery service if the costs for establishing all required capacities within the network becomes too high. In this section, we study the impact of including the express option by varying the express cost. To simulate changes in the express cost, we multiply its value by a coefficient which we vary between 0.2 and 2.

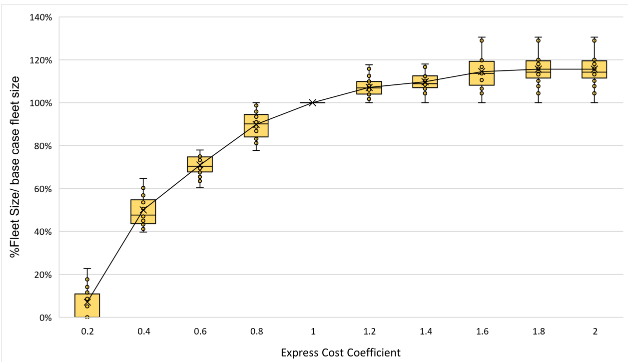


Figure 4.3: *Number of scheduled trucks for different values of the express cost coefficient.*

Figure 4.3 reveals that with high express costs, the decision maker is reluctant to use it as installing a scheduled truck service will be cheaper although its capacity will not be used efficiently (see below). As a result, more scheduled trucks will be installed such that all hubs are connected to the scheduled truck network. However, if express costs are low, the volume shipped via express will increase and scheduled trucks will only be established on the connections where loading rates are very high (up to the point where no scheduled trucks will be installed as they are never competitive against the express service).

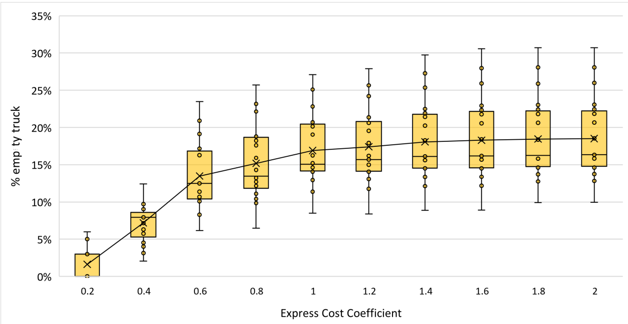


Figure 4.4: *The percentage of unused capacity (empty truckload) for the scheduled truck service.*

The relationship between the capacity utilization of the scheduled truck service and the express cost is visualized specifically in Figure 4.4. In this figure, each observation refers to the average empty capacity of the scheduled trucks for one instance. In the case express costs are too high, and therefore the service is hardly used, we observe that the unused capacity of the scheduled truck service ranges between 10 and 30 percent (on average slightly below 20%). By making the express service more attractive, inefficient scheduled truck transports are replaced by express delivery up to the point where we see a close to 100% capacity utilization ($\geq 95\%$) of all scheduled trucks in the system.

Impact of hub capacity and inventory holding cost

Another unique feature of our model is the consideration of capacity restrictions in the hubs. We expect that the more we restrict the capacity of the hubs, the higher the operational cost will be as it is more likely that shipments will have to deviate from their shortest / cheapest route from source to destination to avoid capacity violations in the hubs.

For each instance, we first determine the *maximum hub capacity* Q^V_{MAX}. This is the minimum capacity for which the capacity constraints become non-binding (i.e., capacity is no longer a constraint in our optimization model and the solution matches the solution with infinite capacity in the hubs). We now run different simulation experiments in which we set the hub capacity equal to a percentage of the maximum hub capacity.

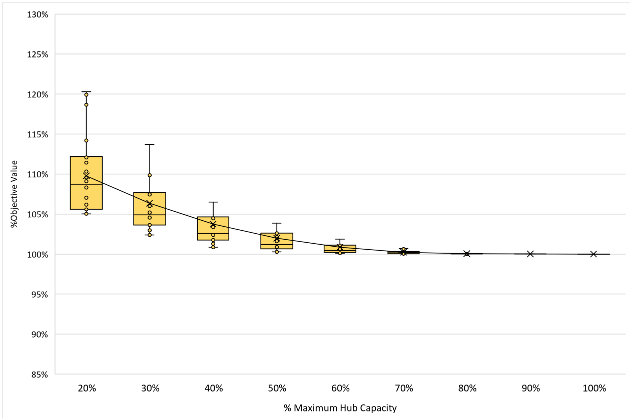


Figure 4.5: *Total network cost as a function of the hub capacity, measured as a percentage of the maximum hub capacity* Q^V_{MAX}.

Figure 4.5 visualises the relationship between Q^V_{MAX} and the total network cost. The base line is given by the scenario with infinite capacity (a very large number). The more we restrict the hub capacity, the higher the total network cost. We see that the total cost increases slightly, with an average cost increase of around 10% if only one fifth of the non-binding hub capacity is available in the network.

Further analysis reveals that this increase is mainly due to an increase in fleet size for the scheduled truck service and a slightly higher utilization of the express delivery option (see Figure 4.6). The reason is twofold. First, the lack of capacity requires shipments to deviate from their shortest path more often. To accommodate these detours, more capacity is required in the scheduled truck service. Second, these detours increase the cost of a shipment when shipped via the scheduled truck service. Consequently, the express delivery option becomes more attractive to cover certain connections.

The fact that the additional truck capacity installed when the hub capacities are very restrictive is corroborated by the relationship between the capacity utilization of the scheduled trucks (measured as the percentage of unused volume) and the maximum hub capacity Q^V_{MAX} given in Figure 4.7. The Figure shows that despite the in-

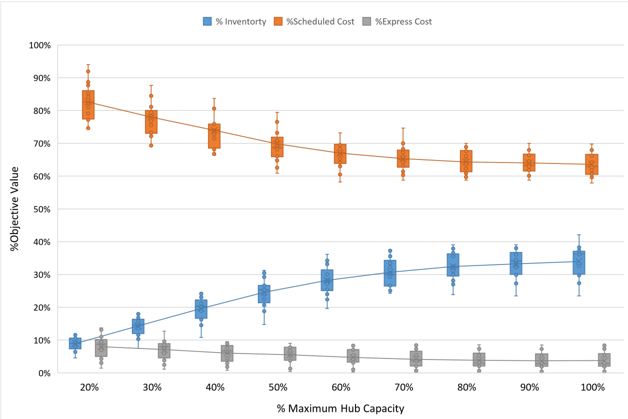


Figure 4.6: *Different cost factors as a function of the hub capacity, measured as a percentage of the maximum hub capacity* Q^V_{MAX}.

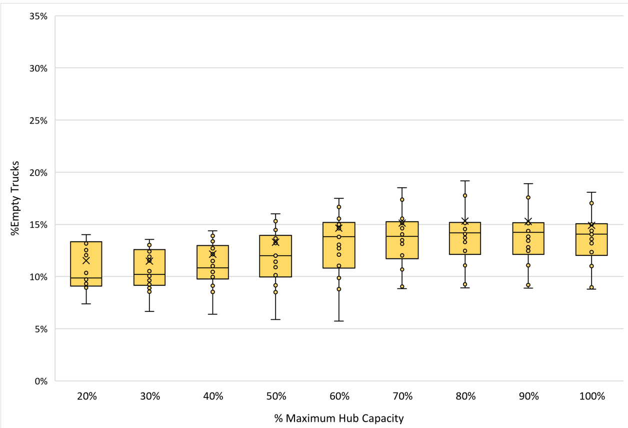
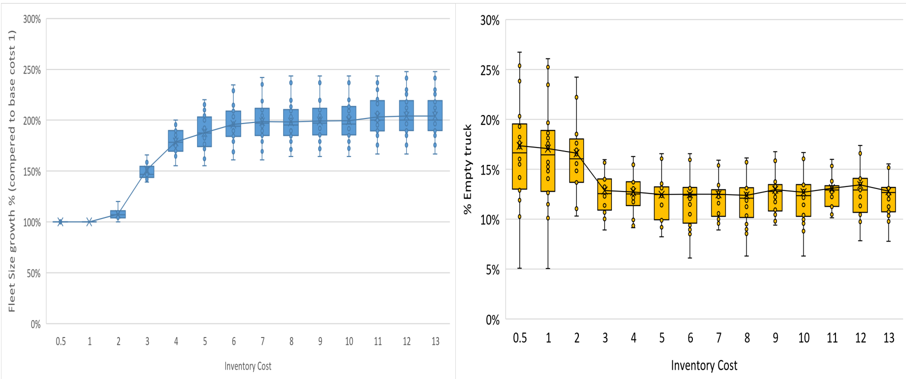


Figure 4.7: *Vehicle utilization of the scheduled truck service as a function of the hub capacity, measured as a percentage of the maximum hub capacity* Q^V_{MAX}.

crease in fleet size, the vehicle utilization increases slightly, from around 85% to close to 90%. This shows that due to limited hub capacities, it becomes more attractive to have the shipments 'stored' during transport. In other words, due to the limited hub capacities, the model is forced to do detours, which requires not only more scheduled trucks, but also longer total transit times between origin and destination (longer path). This automatically leads to more "storage in transit" in order to meet the deadline at the source.

Similar conclusions are found when increasing the inventory holding costs. For increasing values of the holding cost, keeping inventory in the hubs becomes less attractive and more costly. As such, the same decision will be made as when inventory capacity is restricted by the model. We prefer keeping shipments moving on the road by installing a larger fleet of scheduled trucks to bridge the gap between their release time and dispatching time and are willing to accept express deliveries from source to destination more often as no intermediate inventory costs occur then (See figure 4.8).



(a) *Optimal fleet size for different inventory cost*   ***(b)*** *percentage of empty trucks with respect to different inventory cost*

Figure 4.8: *Inventory cost analysis for Canad benchmark cases*

4.6.4 Results for the stochastic multi-period problem variant

Impact of demand variance

To generate the instances, the stochastic demand for shipment k under scenario s was generated based upon a normal distribution as follows,

$$q_{ks} = N(q_k, \alpha q_k)$$

in which q_k represents the average demand of shipment k. The standard deviation is defined as a proportion α of the mean value. In this section, we will vary the variability in the data by changing the value for α within the interval $[0, 0.5]$.

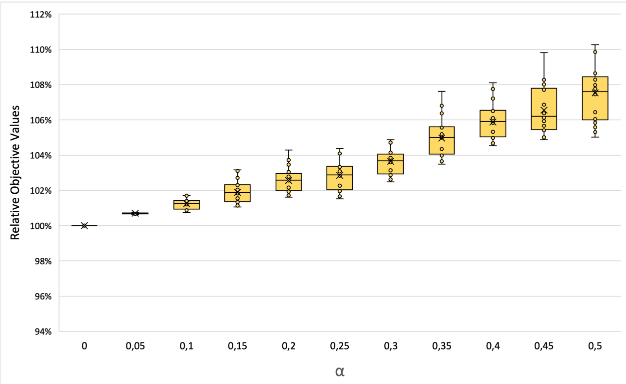
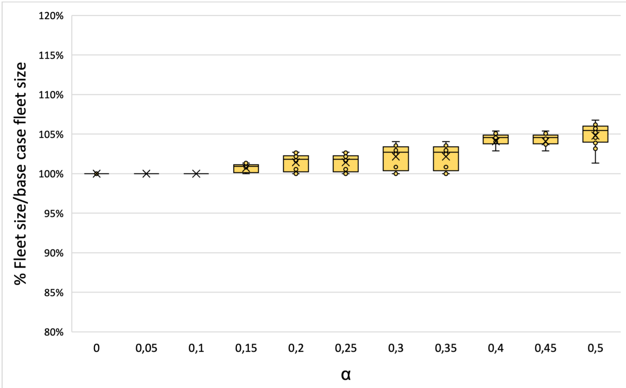


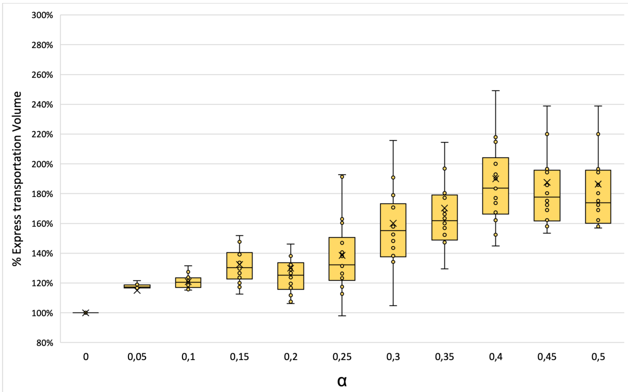
Figure 4.9: Total network cost as a function of the standard deviation of the demand, denoted by α.

The relationship between the total network cost and the value of α is visualized in Figure 4.9. When α equals zero — our baseline scenario (deterministic)— , there is no variability in the demand (i.e., there is only one scenario with the demand for each shipment equal to q_k in each period). As expected, the total network cost grows for increasing values of α, but the increase remains relatively small (with up to 10% cost increase on average for α equal to 0.5).

Investigating the relationship between α and the network configuration, we see that the demand uncertainty mainly impacts the need for express delivery. The volume shipped via express delivery increases fast, even for small values of α. It is nice to see that our simulation results align with the original motivation for considering express delivery as an alternative transport mode. Whereas the scheduled truck service provides a baseline capacity on the links of the network where a considerable flow is guaranteed, express delivery offers the flexibility to absorb the variations above this baseline capacity.



(a) *Optimal fleet size under different standard deviations*



(b) *Expected express transportation volume under different standard deviations*

Figure 4.10: *Commodity variance analysis for Canad benchmark cases*

The Expected Value of Perfect Information (EVPI) and Value of Stochastic Solution (VSS)

In this section, we compare the performance of our two-stage stochastic model with decision making based upon expected values or under perfect information. In decision making based upon expected value, we solve the model only once for a single (average) scenario, i.e., we would consider solely the scenario in which

$$q_k = \frac{1}{|\boldsymbol{\Omega}|} \sum_{s \in \boldsymbol{\Omega}} q_{ks}.$$

To solve the model under perfect information, we first compute the total network cost of each individual scenario. Then, we take the average over all objective function values found as the expected cost under perfect information (recall that we assume each scenario to occur with the same probability).

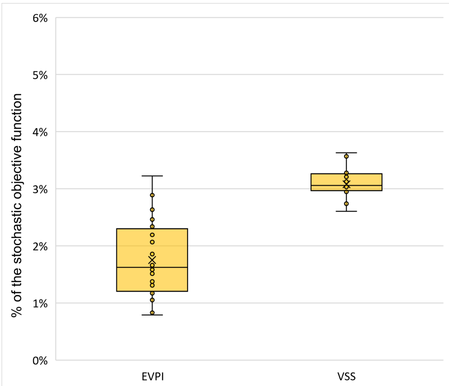


Figure 4.11: *VSS and EVPI Percentage with respect to the objective function value of the stochastic mode*

Figure 4.11 summarizes our main results. Here, we plot the *expected value under perfect information* (EVPI) as well as the *value of stochastic solution* (VSS). Both measures are computed relative to the total network cost obtained when applying the two-stochastic model. The value of perfect information relates to the decrease in total network cost once the decision maker no longer faces uncertainty on the demand for each shipment. In other words, having perfect information will lead to a decrease of 1.8% on average in total network cost.

The value of stochastic solution compares decision making under expected value with the stochastic model. In other words, if the decision maker would ignore the demand variability and solely optimize for the expected values for q_k, the average cost would be around 3.2% higher.

4.7 Conclusion

Motivated by a real case study from the industry, we presented a periodic multi-commodity service network design problem to model the decisions of a 3PL when managing all logistics operations of a supply network using both scheduled truck services (representing long-term agreements with carriers to provide regular capacities on specific network links) and ad-hoc express delivery. Next to the multi-modal approach, we also include the time dimension, hub capacities and account for stochastic demand.

Our computational experiments show that our proposed exact model performs very well. For the small instances with up to 20 hubs and 50 (shipments), the model finds the optimal solution within the time limit. For the average instances with 20-30 hubs and up to 100 commodities, our exact model gives very promising solutions with maximum of 10% optimal gap, and for the large instances (30 hubs and more than 100 commodities) the optimal gap within the time limit is 20% on average.

Adding the option of express delivery as an alternative to the scheduled truck service leads to a lower network cost. This is due to the fact that express deliveries can replace low-volume connections where installing a fixed capacity is not cost-effective. This is similar to passenger transport, where bus services are replaced by on-demand bus lines or taxi rides in rural areas with very low demand.

Furthermore, we show that limiting the available hub capacity increases the fleet size for the scheduled truck service and express delivery. At the same time, it leads to better vehicle utilization for the scheduled truck service. Similar results were found when the inventory holding cost increases. In both scenarios, we observe that the available scheduled truck capacity is used as inventory capacity during transport to bridge the gap between release time and dispatching time. Moreover, adding the express delivery improves the objective function and decreases the total costs in comparison to the case where there exists no express delivery and all the commodities are transported by the scheduled trucks.

We extend the deterministic single-period model to a stochastic multi-period variant in which the variation in demand over the different periods is included explicitly in the model. In contrast to a deterministic case, based on the average demand solely, the inclusion of stochastic demand leads to a 3.2% network cost reduction on average.

As we present an exact solution method, based upon the principles of branch-and-price, we see that the model lacks some scalability towards large (potentially more realistic) instances. The development of an efficient heuristic and sample strategy that allows good convergence to the optimal solution in a short computation time would be valuable.

Further promising extensions of the model left for further research are the addition of a delivery time window (instead of a fixed dispatching time), a heterogeneous fleet for the scheduled truck service (e.g., large trailers vs small(er) vans), social constraints related to the drivers (e.g., breaks, route duration, etc.), and additional sources of uncertainty (e.g. stochastic travel times,etc.). Another itinerary for further research could be the modelling of the pricing decision of the scheduled truck services between the 3PL and the carrier or set of potential carriers each covering a certain part of the envisaged network.

5 Summary

The overarching objective of this thesis is to develop some mathematical models and algorithms for optimizing two challenging and important problems in supply chain management, namely the order picker routing problem in the warehouse and the multi-commodity network design problem. In order to reach this objective, we started with an introduction chapter to understand the fundamental properties of the problems. In the second and third chapter, we studied the order picker routing problem in a standard warehouse layout since warehousing service is a very important component of the logistics system and plays a vital role in the supply chain process. Due to the many new technological advancements such as significant growth in digital marketing and e-commerce, introduction of operating programs such as Just-In-Time (JIT), cycle time reduction and quick response to orders and new marketing strategies such as micro marketing, the number of warehouses worldwide has seen significant growth. One of the processes within warehouses that provide significant increase in the efficiency and cutting the costs while ensuring high customer service levels is the so-called order picking. Order picking is often cited as one of the most critical process among the internal logistics operations due to its massive time and energy requirements. Researchers have already created a variety of exact and heuristic routing methods to reduce the price of order picking. However, the exact algorithms that do model the particularities of the order picking environment are not scalable and they only exist for small warehouses with only two blocks (see Roodbergen and De Koster [136] and Cornuéjols, Fonlupt, and Naddef [39], while for other larger warehouse layouts, some heuristic and meta-heuristic methods are provided. Moreover, the algorithms that solve the problem to optimality, are based on very general assumptions (solve it like a TSP) which leads to inefficient running times that are not applicable in real-life (same day delivery and JIT).

In the second chapter of this thesis, we proposed an exact model that relies on the particularities of a warehouse environment that is better scalable thanks to intensive graph reduction. The average graph reduction in comparison to the complete graph (before pre-processing) is 72.85%, which is really considerable. To the best of our knowledge our exact algorithm is the first proposed algorithm in the literature which can be implemented in warehouses with much more than two blocks and the computation time is comparable and less than one minute. Our mathematical model solves all the instances optimally. Furthermore, we proved that our proposed graph-reduction algorithm in addition to the exact model can be generalized for all planar TSPs, which leads to better performance for a whole set of optimization problems, not only order picker routing problem in the warehouse.

Planar graph reduction methods can be useful to researchers, industry, and society in several ways by designing efficient network topology since it has various applications such as transportation networks, biological network, electronic circuit design, image and video processing, graph theory, social networks, and communication networks and power grids. By reducing the complexity of the network, planar graph reduction methods can lead to simpler and more efficient network designs, and reduce the computational complexity of graph algorithms, which can save time and resources.

In an industrial perspective, our proposed algorithm could be easily integrated in existing warehouse management software, and it would help companies to rely on higher-quality solutions (optimal) for a realistic problem size resulting in a very fast and efficient delivery system. To do so, some additional steps are needed to put our results into practice, such as designing a new software, prototyping, programming, marketing etc.

In the third chapter, we went a step further, considering more general and realistic case of order picking within modern warehouses with scattered storage policies. In industry, many warehouses products are stored at multiple locations as a result of random / scattered storage policies. These policies are used since they are easy to implement, scalable and flexible in handling the seasonality without reserving unnecessary capacity for peak periods of a product. However, in industry, very simplified and basic algorithms are applied in such a picking environment (easier for operators), which is resulting in the routes far from the optimal solution. Our proposed exact GTSP model in addition to the heuristic algorithm could easily be used by warehouse operators and satisfies the current needs by saving time and increasing the efficiency.

In the literature, while the GTSP has received significant attention, the research has primarily focused on non-overlapping clusters. The existing algorithms rely on problem-specific characteristics that assume the clusters to be non-overlapping geographically. In our proposed order picker routing problem, we identified a field of application where this criteria no longer holds. Consequently, the existing algorithms would not be able to exploit this feature and would – likely – perform worse. Our goal was to develop an algorithm that can deal with the clustered substructure of the problem as good as possible, without having to rely on the assumption of non-overlapping clusters. This research was motivated not only by the potential efficiencies that can be achieved in a warehouse, but also by the fact that very little attention has been paid to this problem when accounting for the overlap of clusters. In the third chapter of this thesis, a new problem was introduced and mathematically formulated (GTSP with overlapping clusters), Moreover, we linked it to existing concepts and developed a heuristic algorithm for obtaining high-quality solutions. Our algorithmic ideas can easily be transferred to similar problems in the field of logistics.

Currently the models discussed in the chapters two and three of this thesis are limited to environments with only one picker. However, as a future research, one can consider the possibility to extend our models to multiple order pickers and analyze the impact of the employment of multiple order pickers on the average processing time of the orders. In this case we might consider the optimal number of pickers and order assignment policies to each order picker. Another approach could be assigning each order picker to a specific zone in the warehouse and try to assign them to only part of the items in each order, located in their zone. In addition to this modifications, various promising areas for further research can be identified in the field of order picker routing problem. One of these modifications to the current study might be taking into account numerous deposit locations (depots) and investigating its effects on tour lengths. Moreover, due dates for order list, dynamic customer orders, development of problem-specific solution procedures, and inclusion of uncertain expected orders should also be taken into account in the decision-making process. Furthermore, for chapter three of the thesis, future research could also focus on the pre-processing of the graph and developing a graph reduction method in the presence of overlapping clusters.

In the fourth chapter of this thesis, we consider a variant of multi-modal Multi-commodity Network design problem with delivery time and stochastic demand. The problem presented in this chapter is motivated by a case study in which a 3PL is responsible for coordinating all material flows that belong to the supply network of a large construction company within Europe (company names are confidential). This study contributes to the academic literature in the following ways. First, we consider multi-commodity service network design models over time to allow differentiation between the (periodic) scheduled truck services and the ad-hoc express delivery option. Moreover, we consider the delivery date for commodities, and finally fleet management by determining the allocation and fleet size of each type of transportation mode at each hub for each time period. These assumptions make our model more realistic. Second, we account for potential capacity limitations in the hub and manage inventory levels accordingly. Furthermore, we develop competitive solution approaches based on an integration of column generation and branch-and-price to solve this realistic variant of the MCND. Moreover, we extend our models and results to a setting with uncertain demand and present a two-stage, scenario-based stochastic model which is solved using the sample average approximation method. Finally, a broad range of managerial insights have been generated by means of an extensive sensitivity analysis.

Future contributions for this study can include considering heterogeneous vehicles with different capacities, improving the solution method (column generation and branch and price), taking into account the multi-period planning since our proposed model is focused on a single period with multiple time intervals. Moreover, robust optimization with service levels and time window for commodity delivery could be other future research directions.

6
Impact of the Thesis

In this thesis we study two lines of work, first the order picker routing problem in a warehouse, and second, the multi-commodity network design problem. Both problems play a pivotal role in shaping operational efficiency, customer satisfaction, cost reduction, and ultimately, an enhanced supply chain performance.

From an economical perspective, using the proposed models and algorithms can improve the efficiency of warehouse and distribution center operations, which can contribute to economic growth and development. Another economical outcome of this thesis is that the proposed models can result in an increased competitiveness for different logistic businesses This can benefit society by promoting innovation and economic growth. One of the social benefits of this thesis is improving the working conditions for warehouse and distribution center employees. This can improve job satisfaction and employee retention. Moreover, the proposed models in this thesis can result in a better resource allocation in warehouses and distribution centers which can help organizations allocate resources more effectively and make better use of their labor force.

As a take-away from this thesis, our proposed models and algorithms in the field of supply chain optimization can bring benefits to researchers by helping them working on these problems to develop new mathematical models, algorithms, and software tools to solve them. These innovations can lead to improvements in supply chain management and logistics, as well as new business opportunities for companies in the industry. In an industrial perspective, the current research can help improving the efficiency of warehouse and distribution center operations, resulting in reduced costs, improved delivery times, and increased customer satisfaction. This can benefit both the industry and society by increasing customer loyalty and promoting economic growth.

Moreover, improving supply chain performance in the fields of order picking and service network design can result in increased industry profitability, enhanced customer satisfaction, and a market advantage. Moreover, our proposed algorithm could be easily integrated in existing warehouse management software, and it would help companies to rely on higher-quality solutions (optimal) for a realistic problem size resulting in a very fast and efficient delivery system. To do so, some additional steps are needed to put our results into practice, such as designing a new software, prototyping, programming, marketing etc.

In addition to all the results and contributions to the literature, achieved by this thesis, our approaches for the studied problems are versatile in the sense that they can be used in combination with other approaches to help improve the further research or initiate further research on their respective problems.

Bibliography

[1] Amir Reza Ahmadi Keshavarz, Davood Jaafari, Mehran Khalaj, and Parshang Dokouhaki. "A survey of the literature on order-picking systems by combining planning problems". In: *Applied Sciences* 11.22 (2021), page 10641.

[2] Shabbir Ahmed, Alexander Shapiro, and Er Shapiro. "The sample average approximation method for stochastic programs with integer recourse". In: *Submitted for publication* (2002), pages 1–24.

[3] M Hakan Akyüz, Temel Öncan, and İ Kuban Altınel. "Branch and bound algorithms for solving the multi-commodity capacitated multi-facility Weber problem". In: *Annals of Operations Research* 279.1 (2019), pages 1–42.

[4] Michael Omotayo Alabi and Ojelanki Ngwenyama. "Food security and disruptions of the global food supply chains during COVID-19: building smarter food supply chains for post COVID-19 era". In: *British Food Journal* (2022).

[5] Abdullah Alsheddy, Christos Voudouris, Edward PK Tsang, and Ahmad Alhindi. *Guided Local Search.* 2018.

[6] Jardar Andersen, Marielle Christiansen, Teodor Gabriel Crainic, and Roar Grønhaug. "Branch and price for service network design with asset management constraints". In: *Transportation Science* 45.1 (2011), pages 33–49.

[7] Ehsan Ardjmand, Omid Sanei Bajgiran, and Eyad Youssef. "Using list-based simulated annealing and genetic algorithm for order batching and picker routing in put wall based picking systems". In: *Applied Soft Computing* 75 (2019), pages 106–119.

[8] Ehsan Ardjmand, Heman Shakeri, Manjeet Singh, and Omid Sanei Bajgiran. "Minimizing order picking makespan with multiple pickers in a wave picking warehouse". In: *International Journal of Production Economics* 206 (2018), pages 169–183.

[9] Sanjeev Arora. "Approximation schemes for NP-hard geometric optimization problems: A survey". In: *Mathematical Programming* 97 (July 2003), pages 43–69. DOI: 10.1007/s10107-003-0438-y.

[10] Abbas Azadi Moghaddam Arani, Fariborz Jolai, and Mohammad Mahdi Nasiri. "A multi-commodity network flow model for railway capacity optimization in case of line blockage". In: *International Journal of Rail Transportation* 7.4 (2019), pages 297–320.

[11] Amir Hossein Azadnia, Shahrooz Taheri, Pezhman Ghadimi, Muhamad Zameri Mat Saman, and Kuan Yew Wong. "Order batching in warehouses by minimizing total tardiness: a hybrid approach of weighted association rule mining and genetic algorithms". In: *The Scientific World Journal* 2013 (2013).

[12] Hossein Badri, SMT Fatemi Ghomi, and Taha-Hossein Hejazi. "A two-stage stochastic programming approach for value-based closed-loop supply chain network design". In: *Transportation Research Part E: Logistics and Transportation Review* 105 (2017), pages 1–17.

[13] Martin B Bagaram and Sándor F Tóth. "Multistage sample average approximation for harvest scheduling under climate uncertainty". In: *Forests* 11.11 (2020), page 1230.

[14] A Balakrishnan, TL Magnanti, and P Mirchandani. *Annotated bibliographies in combinatorial optimization, chap. 18, Network Design*. 1997.

[15] Pouya Baniasadi, Mehdi Foumani, Kate Smith-Miles, and Vladimir Ejov. "A transformation technique for the clustered generalized traveling salesman problem with applications to logistics". In: *European Journal of Operational Research* 285.2 (2020), pages 444–457.

[16] Gulay Barbarosoğlu and Yasemin Arda. "A two-stage stochastic programming framework for transportation planning in disaster response". In: *Journal of the operational research society* 55.1 (2004), pages 43–53.

[17] Ramin Bazrafshan, Sarfaraz Zolfani, and S.M.J. Mirzapour Al-e-hashem. "Comparison of the Sub-Tour Elimination Methods for the Asymmetric Traveling Salesman Problem Applying the SECA Method". In: *Axioms* 10 (Feb. 2021), page 19. DOI: 10.3390/axioms10010019.

[18] Tolga Bektaş, Güneş Erdoğan, and Stefan Røpke. "Formulations and branch-and-cut algorithms for the generalized vehicle routing problem". In: *Transportation Science* 45.3 (2011), pages 299–316.

[19] Nicolas Belanger, Guy Desaulniers, François Soumis, and Jacques Desrosiers. "Periodic airline fleet assignment with time windows, spacing constraints, and time dependent revenues". In: *European Journal of Operational Research* 175.3 (2006), pages 1754–1766.

[20] David Ben-Arieh, Gregory Gutin, M Penn, Anders Yeo, and Alexey Zverovitch. "Transformations of generalized ATSP into ATSP". In: *Operations Research Letters* 31.5 (2003), pages 357–365.

[21] Maurizio Boccia, Teodor Gabriel Crainic, Antonio Sforza, and Claudio Sterle. "Multi-commodity location-routing: Flow intercepting formulation and branch-and-cut algorithm". In: *Computers & Operations Research* 89 (2018), pages 94–112.

[22] Tamás Bódis and János Botzheim. "Bacterial memetic algorithms for order picking routing problem with loading constraints". In: *Expert Systems with Applications* 105 (2018), pages 196–220.

[23] Boris Bontoux, Christian Artigues, and Dominique Feillet. "A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem". In: *Computers & Operations Research* 37.11 (2010), pages 1844–1852.

[24] Ozan Çakır. "Benders decomposition applied to multi-commodity, multi-mode distribution planning". In: *Expert Systems with Applications* 36.4 (2009), pages 8212–8217.

[25] Hadrien Cambazard and Nicolas Catusse. "Fixed-parameter algorithms for rectilinear steiner tree and rectilinear traveling salesman problem in the plane". In: *European Journal of Operational Research* 270.2 (2018), pages 419–429.

[26] Giovanni Campuzano, Carlos Obreque, and Maichel M Aguayo. "Accelerating the Miller–Tucker–Zemlin model for the asymmetric traveling salesman problem". In: *Expert Systems with Applications* 148 (2020), page 113229.

[27] Cem Canel, Basheer M Khumawala, Japhett Law, and Anthony Loh. "An algorithm for the capacitated, multi-commodity multi-period facility location problem". In: *Computers & Operations Research* 28.5 (2001), pages 411–427.

[28] Jose Alejandro Cano, Alexander Alberto Correa-Espinal, and Rodrigo Andrés Gómez-Montoya. "An evaluation of picking routing policies to improve warehouse efficiency". In: *International Journal of Industrial Engineering and Management* 8.4 (2017), pages 229–238.

[29] Angelo Castelda. *Understanding The Impacts of eCommerce On Warehouse Operations*. `https://www.floship.com/blog/_ecommerce-warehouse-operations/`. Accessed: 2020-05-27.

[30] Melih Celik and Haldun Sural. "The order picking problem in fishbone aisle warehouses". In: (2012).

[31] Melih Celik and HALDUN Süral. "Order picking in a parallel-aisle warehouse with turn penalties". In: *International Journal of Production Research* 54.14 (2016), pages 4340–4355.

[32] Melih Çelik and H Süral. "Order picking in a parallel-aisle warehouse with turn penalties". In: *International Journal of Production Research* 54.14 (2016), pages 4340–4355.

[33] Thomas Chabot, Rahma Lahyani, Leandro C Coelho, and Jacques Renaud. "Order picking problems under weight, fragility and category constraints". In: *International Journal of Production Research* 55.21 (2017), pages 6361–6379.

[34] Fangyu Chen, Hongwei Wang, Yong Xie, and Chao Qi. "An ACO-based online routing method for multiple order pickers with congestion consideration in warehouse". In: *Journal of Intelligent Manufacturing* 27.2 (2016), pages 389–408.

[35] Fangyu Chen, Gangyan Xu, and Yongchang Wei. "Heuristic routing methods in multiple-block warehouses with ultra-narrow aisles and access restriction". In: *International Journal of Production Research* 57.1 (2019), pages 228–249.

[36] Tzu-Li Chen, Chen-Yang Cheng, Yin-Yann Chen, and Li-Kai Chan. "An efficient hybrid algorithm for integrated order batching, sequencing and routing problem". In: *International Journal of Production Economics* 159 (2015), pages 158–167.

[37] Martin Christopher. "The mitigation of risk in resilient supply chains". In: (2018).

[38] Jean-François Cordeau, Gilbert Laporte, Martin WP Savelsbergh, and Daniele Vigo. "Vehicle routing". In: *Handbooks in operations research and management science* 14 (2007), pages 367–428.

[39] Gérard Cornuéjols, Jean Fonlupt, and Denis Naddef. "The traveling salesman problem on a graph and some related integer polyhedra". In: *Mathematical programming* 33.1 (1985), pages 1–27.

[40] Pablo Cortés, Rodrigo A Gómez-Montoya, Jesús Muñuzuri, and Alexander Correa-Espinal. “A tabu search approach to solving the picking routing problem for large-and medium-size distribution centres considering the availability of inventory and K heterogeneous material handling equipment”. In: *Applied Soft Computing* 53 (2017), pages 61–73.

[41] Teodor G Crainic and Jean-Marc Rousseau. “Multicommodity, multimode freight transportation: A general modeling and algorithmic framework for the service network design problem”. In: *Transportation Research Part B: Methodological* 20.3 (1986), pages 225–242.

[42] Teodor Gabriel Crainic, Pierre Dejax, and Louis Delorme. “Models for multimode multicommodity location problems with interdepot balancing requirements”. In: *Annals of Operations Research* 18.1 (1989), pages 277–302.

[43] Teodor Gabriel Crainic, Antonio Frangioni, and Bernard Gendron. “Bundle-based relaxation methods for multicommodity capacitated fixed charge network design”. In: *Discrete Applied Mathematics* 112.1-3 (2001), pages 73–99.

[44] Marco Croucamp and Jacomine Grobler. “Metaheuristics for the robot part sequencing and allocation problem with collision avoidance”. In: *Progress in Artificial Intelligence: 20th EPIA Conference on Artificial Intelligence, EPIA 2021, Virtual Event, September 7–9, 2021, Proceedings 20*. Springer. 2021, pages 469–481.

[45] George B Dantzig. “Linear programming under uncertainty”. In: *Management science* 1.3-4 (1955), pages 197–206.

[46] René De Koster, Tho Le-Duc, and Kees Jan Roodbergen. “Design and control of warehouse order picking: A literature review”. In: *European journal of operational research* 182.2 (2007), pages 481–501.

[47] René De Koster and Edo Van Der Poort. “Routing orderpickers in a warehouse: a comparison between optimal and heuristic solutions”. In: *IIE transactions* 30.5 (1998), pages 469–480.

[48] Roberta De Santis, Roberto Montanari, Giuseppe Vignali, and Eleonora Bottani. “An adapted ant colony optimization algorithm for the minimization of the travel distance of pickers in manual warehouses”. In: *European Journal of Operational Research* 267.1 (2018), pages 120–137.

[49] Izabela Dembińska. "The Impact of E-Commerce Development on the Warehouse Space Market in Poland". In: *Economics and culture* 13.2 (2016), pages 5–13.

[50] M Devaraj. *Impact of eCommerce Growth on the Logistics Sector*. https://www.linkedin.com/pulse/impact-ecommerce-growth-logistics-sector-devaraj-m/?trk=pulse-article_more-articles_related-content-card. Accessed: 2022-06-23.

[51] Mary Dillon, Fabricio Oliveira, and Babak Abbasi. "A two-stage stochastic programming model for inventory management in the blood supply chain". In: *International Journal of Production Economics* 187 (2017), pages 27–41.

[52] Scott DuHadway, Steven Carnovale, and Benjamin Hazen. "Understanding risk management for intentional supply chain disruptions: Risk detection, risk mitigation, and risk recovery". In: *Annals of Operations Research* 283.1 (2019), pages 179–198.

[53] Goran Dukic and Cedomir Oluic. "Order-picking methods: improving order-picking efficiency". In: *International Journal of Logistics Systems and Management* 3.4 (2007), pages 451–460.

[54] Mehdi El Krari, Belaïd Ahiod, El Benani, and Youssef Bouazza. "A preprocessing reduction method for the generalized travelling salesman problem". In: *Operational Research* 21.4 (2021), pages 2543–2591.

[55] Judith M Farvolden and Warren B Powell. "Subgradient methods for the service network design problem". In: *Transportation Science* 28.3 (1994), pages 256–272.

[56] Matteo Fischetti, Juan José Salazar González, and Paolo Toth. "A branch-and-cut algorithm for the symmetric generalized traveling salesman problem". In: *Operations Research* 45.3 (1997), pages 378–394.

[57] Bernhard Fleischmann. "A cutting plane procedure for the travelling salesman problem on road networks". In: *European Journal of Operational Research* 21.3 (1985), pages 307–317.

[58] Xuehong Gao and Gyu M Lee. "A stochastic programming model for multi-commodity redistribution planning in disaster response". In: *IFIP international conference on advances in production management systems*. Springer. 2018, pages 67–78.

[59] M. R. Garey, R. L. Graham, and D. S. Johnson. “Some NP-Complete Geometric Problems”. In: *Proceedings of the Eighth Annual ACM Symposium on Theory of Computing*. STOC ’76. Hershey, Pennsylvania, USA: Association for Computing Machinery, 1976, pages 10–22. DOI: 10.1145/800113.803626.

[60] Manfred Gilli, Dietmar Maringer, and Enrico Schumann. *Numerical methods and optimization in finance*. Academic Press, 2019.

[61] Christoph H Glock and Eric H Grosse. “Storage policies and order picking strategies in U-shaped order-picking systems with a movable base”. In: *International Journal of Production Research* 50.16 (2012), pages 4344–4357.

[62] Marc Goetschalckx and H Donald Ratliff. “Order picking in an aisle”. In: *IIE transactions* 20.1 (1988), pages 53–62.

[63] Rodrigo Andrés Gómez-Montoya, Alexander Alberto Correa-Espinal, and José Daniel Hernández-Vahos. “Picking Routing Problem with K homogenous material handling equipment for a refrigerated warehouse”. In: *Revista Facultad de Ingeniería Universidad de Antioquia* 80 (2016), pages 9–20.

[64] David B Grant, Chee Yew Wong, and Alexander Trautrims. *Sustainable logistics and supply chain management: principles and practices for sustainable operations and management*. Kogan Page Publishers, 2017.

[65] Beth Gutelius, Nik Theodore, et al. “The future of warehouse work: Technological change in the US logistics industry”. In: *UC Berkeley Labor Center* (2019).

[66] Gregory Gutin and Daniel Karapetyan. “A memetic algorithm for the generalized traveling salesman problem”. In: *Natural Computing* 9.1 (2010), pages 47–60.

[67] Gregory Gutin and Daniel Karapetyan. “Generalized traveling salesman problem reduction algorithms”. In: *Algorithmic Operations Research* 4.2 (2009), pages 144–154.

[68] Randolph W Hall. “Distance approximations for routing manual pickers in a warehouse”. In: *IIE transactions* 25.4 (1993), pages 76–87.

[69] Bayan Hamdan and Ali Diabat. “A two-stage multi-echelon stochastic blood supply chain problem”. In: *Computers & Operations Research* 101 (2019), pages 130–143.

[70] Erik Hellsten, David Franz Koza, Ivan Contreras, Jean-François Cordeau, and David Pisinger. "The transit time constrained fixed charge multicommodity network design problem". In: *Computers & Operations Research* 136 (2021), page 105511.

[71] Keld Helsgaun. "General k-opt submoves for the Lin–Kernighan TSP heuristic". In: *Mathematical Programming Computation* 1.2 (2009), pages 119–163.

[72] Keld Helsgaun. "Solving the equality generalized traveling salesman problem using the Lin–Kernighan–Helsgaun algorithm". In: *Mathematical Programming Computation* 7.3 (2015), pages 269–287.

[73] AL Henry-Labordere. "The record balancing problem: A dynamic programming solution of a generalized traveling salesman problem". In: *Revue Francaise D Informatique DeRecherche Operationnelle* 3.2 (1969), pages 43–49.

[74] SS Ho and S Sarma. "The fragmented warehouse: Location assignment for unit-load picking". In: *2008 IEEE International Conference on Industrial Engineering and Engineering Management*. IEEE. 2008, pages 1159–1163.

[75] Y-C Ho and Y-Y Tseng. "A study on order-batching methods of order-picking in a distribution centre with two cross-aisles". In: *International Journal of Production Research* 44.17 (2006), pages 3391–3417.

[76] Jeffrey Hoeft and Udatta S Palekar. "Heuristics for the plate-cutting traveling salesman problem". In: *IIE transactions* 29.9 (1997), pages 719–731.

[77] Ling-feng Hsieh and Lihui Tsai. "The optimum design of a warehouse system on order picking efficiency". In: *The International Journal of Advanced Manufacturing Technology* 28 (2006), pages 626–637.

[78] Bin Hu and Günther R Raidl. "Effective neighborhood structures for the generalized traveling salesman problem". In: *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer. 2008, pages 36–47.

[79] Stefan Irnich, Paolo Toth, and Daniele Vigo. "Chapter 1: The family of vehicle routing problems". In: *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. SIAM, 2014, pages 1–33.

[80] Chawalit Jeenanunta, Boontariga Kasemsontitum, and Tawinan Noichawee. "A multi-commodity flow approach for aircraft routing and maintenance problem". In: *2011 IEEE International Conference on Quality and Reliability*. IEEE. 2011, pages 150–155.

[81] Zhang Kai and Gao Chuanhou. "Improved formulations of the joint order batching and picker routing problem". In: *arXiv preprint arXiv:2207.05305* (2022).

[82] Imdat Kara, Huseyin Guden, and Ozge N Koc. "New formulations for the generalized traveling salesman problem". In: *Proceedings of the 6th international conference on applied mathematics, simulation, modelling, ASM*. Volume 12. 2012, pages 60–65.

[83] Daniel Karapetyan and Gregory Gutin. "Lin–Kernighan heuristic adaptations for the generalized traveling salesman problem". In: *European Journal of Operational Research* 208.3 (2011), pages 221–232.

[84] Naoto Katayama. "A combined capacity scaling and local branching approach for capacitated multi-commodity network design problem". In: *Far East Journal of Applied Mathematics* 92.1 (2015), pages 1–30.

[85] Sujin Kim, Raghu Pasupathy, and Shane G Henderson. "A guide to sample average approximation". In: *Handbook of simulation optimization* (2015), pages 207–243.

[86] Natalio Krasnogor, Giuseppe Nicosia, Mario Pavone, David Pelta, et al. *Nature inspired cooperative strategies for optimization (nicso 2007)*. Volume 129. Springer, 2008.

[87] Kanokporn Kungwalsong, Chen-Yang Cheng, Chumpol Yuangyai, and Udom Janjarassuk. "Two-stage stochastic program for supply chain network design under facility disruptions". In: *Sustainability* 13.5 (2021), page 2596.

[88] Maher Lahmar. *Facility logistics: Approaches and solutions to next generation challenges*. CRC Press, 2007.

[89] Gilbert Laporte, Ardavan Asef-Vaziri, and Chelliah Sriskandarajah. "Some applications of the generalized travelling salesman problem". In: *Journal of the Operational Research Society* 47.12 (1996), pages 1461–1467.

[90] Gilbert Laporte, Luís Lopes, and François Soumis. "Optimal sequencing rules for some large-scale flexible manufacturing problems under the Manhattan and Chebychev metrics". In: *International journal of flexible manufacturing systems* 10 (1998), pages 27–42.

[91] Gilbert Laporte, Hélène Mercure, and Yves Nobert. "Generalized travelling salesman problem through n sets of nodes: the asymmetrical case". In: *Discrete Applied Mathematics* 18.2 (1987), pages 185–197.

[92] Gilbert Laporte and Yves Nobert. "Generalized travelling salesman problem through n sets of nodes: an integer programming approach". In: *INFOR: Information Systems and Operational Research* 21.1 (1983), pages 61–75.

[93] Gilbert Laporte and Frédéric Semet. "Computational evaluation of a transformation procedure for the symmetric generalized traveling salesman problem". In: *INFOR: Information Systems and Operational Research* 37.2 (1999), pages 114–120.

[94] Bernd Lauterbach, Rüdiger Fritz, Jens Gottlieb Gottlieb, Bernd Mosbrucker, and Till Dengel. *Transportation management with SAP TM*. Galileo Press Bonn, 2009.

[95] Adam N Letchford, Saeideh D Nasiri, and Dirk Oliver Theis. "Compact formulations of the Steiner traveling salesman problem and related problems". In: *European Journal of Operational Research* 228.1 (2013), pages 83–92.

[96] Jianbin Li, Rihuan Huang, and James B Dai. "Joint optimisation of order batching and picker routing in the online retailer's warehouse in China". In: *International Journal of Production Research* 55.2 (2017), pages 447–461.

[97] Yao-Nan Lien, Eva Ma, and Benjamin W-S Wah. "Transformation of the generalized traveling-salesman problem into the standard traveling-salesman problem". In: *Information Sciences* 74.1-2 (1993), pages 177–189.

[98] Frauke Liers, Alexander Martin, and Susanne Pape. *Steiner trees with degree constraints: Structural results and an exact solution approach*. Technical report. Technical report, Department Mathematik, 2014.

[99] Chun-Cheng Lin, Jia-Rong Kang, Chung-Chih Hou, and Chen-Yang Cheng. "Joint order batching and picker Manhattan routing problem". In: *Computers & Industrial Engineering* 95 (2016), pages 164–174.

[100] Shen Lin. "Computer solutions of the traveling salesman problem". In: *Bell System Technical Journal* 44.10 (1965), pages 2245–2269.

[101] Shen Lin and Brian W Kernighan. "An effective heuristic algorithm for the traveling-salesman problem". In: *Operations research* 21.2 (1973), pages 498–516.

[102] Zhiyuan Lin and Raymond SK Kwan. "An integer fixed-charge multicommodity flow (FCMF) model for train unit scheduling". In: *Electronic Notes in Discrete Mathematics* 41 (2013), pages 165–172.

[103] Nils Löhndorf. "An empirical analysis of scenario generation methods for stochastic optimization". In: *European Journal of Operational Research* 255.1 (2016), pages 121–132.

[104] Wenrong Lu, Duncan McFarlane, Vaggelis Giannikas, and Quan Zhang. "An algorithm for dynamic order-picking in warehouse operations". In: *European Journal of Operational Research* 248.1 (2016), pages 107–122.

[105] T Magnanti. "L., and Wong R". In: *T.: Network Design and Transportation Planning: Model and Algorithm Transportation Science* 18 (1984).

[106] Binod Maharjan and Timothy I Matis. "Multi-commodity flow network model of the flight gate assignment problem". In: *Computers & Industrial Engineering* 63.4 (2012), pages 1135–1144.

[107] Makusee Masae, Christoph H Glock, and Eric H Grosse. "Order picker routing in warehouses: A systematic literature review". In: *International Journal of Production Economics* 224 (2020), page 107564.

[108] Makusee Masae, Christoph H Glock, and Panupong Vichitkunakorn. "Optimal order picker routing in the chevron warehouse". In: *IISE Transactions* 52.6 (2020), pages 665–687.

[109] Marek Matusiak, René De Koster, and Jari Saarinen. "Utilizing individual picker skills to improve order batching in a warehouse". In: *European Journal of Operational Research* 263.3 (2017), pages 888–899.

[110] Marek Matusiak, René de Koster, Leo Kroon, and Jari Saarinen. "A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse". In: *European Journal of Operational Research* 236.3 (2014), pages 968–977.

[111] Richard D McBride and John W Mamer. "Solving the undirected multicommodity flow problem using a shortest path-based pricing algorithm". In: *Networks: An International Journal* 38.4 (2001), pages 181–188.

[112] Jorge E Mendoza, Bruno Castanier, Christelle Guéret, Andrés Medaglia, and Nubia Velasco. "A simulation-based MOEA for the multi-compartment vehicle routing problem with stochastic demands". In: *8th Metaheuristics International Conference (MIC)*. 2009.

[113] Borja Menéndez, Eduardo G Pardo, Antonio Alonso-Ayuso, Elisenda Molina, and Abraham Duarte. "Variable neighborhood search strategies for the order batching problem". In: *Computers & Operations Research* 78 (2017), pages 500–512.

[114] Clair E Miller, Albert W Tucker, and Richard A Zemlin. "Integer programming formulation of traveling salesman problems". In: *Journal of the ACM (JACM)* 7.4 (1960), pages 326–329.

[115] Michel Minoux. "Discrete cost multicommodity network optimization problems and exact solution methods". In: *Annals of operations research* 106.1 (2001), pages 19–46.

[116] Michel Minoux. "Networks synthesis and optimum network design problems: Models, solution methods and applications". In: *Networks* 19.3 (1989), pages 313–360.

[117] Siamak Moradi, Andrea Raith, and Matthias Ehrgott. "A bi-objective column generation algorithm for the multi-commodity minimum cost flow problem". In: *European Journal of Operational Research* 244.2 (2015), pages 369–378.

[118] Joëlle Morana. *Sustainable supply chain management*. John Wiley & Sons, 2013.

[119] Mahdi Naderi-Beni, Ehsan Ghobadian, Sadoullah Ebrahimnejad, and Reza Tavakkoli-Moghaddam. "Fuzzy bi-objective formulation for a parallel machine scheduling problem with machine eligibility restrictions and sequence-dependent setup times". In: *International Journal of Production Research* 52.19 (2014), pages 5799–5822.

[120] Olegs Nalivajevs and Daniel Karapetyan. "Conditional Markov Chain Search for the Generalised Travelling Salesman Problem for Warehouse Order Picking". In: *2019 11th Computer Science and Electronic Engineering (CEEC)*. IEEE. 2019, pages 75–78.

[121] Charles E Noon and James C Bean. "A Lagrangian based approach for the asymmetric generalized traveling salesman problem". In: *Operations Research* 39.4 (1991), pages 623–632.

[122] Charles E Noon and James C Bean. "An efficient transformation of the generalized traveling salesman problem". In: *INFOR: Information Systems and Operational Research* 31.1 (1993), pages 39–44.

[123] Murat Oğuz, Tolga Bektaş, and Julia A Bennell. “Multicommodity flows and Benders decomposition for restricted continuous location problems”. In: *European Journal of Operational Research* 266.3 (2018), pages 851–863.

[124] Johan Oscar Ong and Don Thomas Joseph. “A review of order picking improvement methods”. In: *J@ ti Undip: Jurnal Teknik Industri* 9.3 (2014), pages 135–138.

[125] Lucie Pansart, Nicolas Catusse, and Hadrien Cambazard. “Exact algorithms for the order picking problem”. In: *Computers & Operations Research* 100 (2018), pages 117–127.

[126] CG Petersen. “Routeing and storage policy interaction in order picking operations”. In: *Decision Sciences Institute Proceedings*. Volume 3. 1995, pages 1614–1616.

[127] Charles G Petersen. “An evaluation of order picking routeing policies”. In: *International Journal of Operations & Production Management* 17.11 (1997), pages 1098–1111.

[128] Charles G Petersen and Gerald Aase. “A comparison of picking, storage, and routing policies in manual order picking”. In: *International Journal of Production Economics* 92.1 (2004), pages 11–19.

[129] Ulrich Pferschy and Joachim Schauer. “Order batching and routing in a non-standard warehouse”. In: *Electronic Notes in Discrete Mathematics* 69 (2018), pages 125–132.

[130] Ragheb Rahmaniani, Teodor Gabriel Crainic, Michel Gendreau, and Walter Rei. “Accelerating the Benders decomposition method: Application to stochastic network design problems”. In: *SIAM Journal on Optimization* 28.1 (2018), pages 875–903.

[131] Farzaneh Rajabighamchi, Stan van Hoesel, and Christof Defryn. *Bus service for cargo: The periodic service network design problem with regular and express deliveries under demand uncertainty*. English. WorkingPaper 005. Netherlands: Maastricht University, Graduate School of Business and Economics, May 2023. DOI: 10.26481/umagsb.2023005.

[132] Farzaneh Rajabighamchi, Stan van Hoesel, and Christof Defryn. *Graph reduction for the planar Travelling Salesman Problem: An application in order picking*. English. WorkingPaper 004. Netherlands: Maastricht University, Graduate School of Business and Economics, May 2023. DOI: 10.26481/umagsb.2023004.

[133] Farzaneh Rajabighamchi, Stan van Hoesel, and Christof Defryn. *The order picking problem under a scattered storage policy*. English. Working-Paper 006. Netherlands: Maastricht University, Graduate School of Business and Economics, May 2023. DOI: 10.26481/umagsb.2023006.

[134] H Donald Ratliff and Arnon S Rosenthal. “Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem”. In: *Operations Research* 31.3 (1983), pages 507–521.

[135] Kees Jan Roodbergen. “Storage assignment for order picking in multiple-block warehouses”. In: *Warehousing in the global supply chain*. Springer, 2012, pages 139–155.

[136] Kees Jan Roodbergen and René De Koster. “Routing order pickers in a warehouse with a middle aisle”. In: *European Journal of Operational Research* 133.1 (2001), pages 32–43.

[137] Kees-Jan Roodbergen. *Layout and routing methods for warehouses*. EPS-2001-004-LIS. 2001.

[138] KJ Roodbergen and R De Koster. “Routing order pickers in a warehouse with multiple cross aisles”. In: *Progress in Material Handling Research* 1998 (1998), pages 451–467.

[139] Andreas Rudi, Magnus Fröhling, Konrad Zimmer, and Frank Schultmann. “Freight transportation planning considering carbon emissions and in-transit holding costs: a capacitated multi-commodity network flow model”. In: *EURO Journal on Transportation and Logistics* 5.2 (2016), pages 123–160.

[140] Alan Rushton, Phil Croucher, and Peter Baker. *The handbook of logistics and distribution management: Understanding the supply chain*. Kogan Page Publishers, 2022.

[141] Sonja M Russo, Jana Voegl, and Patrick Hirsch. “A multi-method approach to design urban logistics hubs for cooperative use”. In: *Sustainable Cities and Society* 69 (2021), page 102847.

[142] Khodakaram Salimifard and Sara Bigharaz. “The multicommodity network flow problem: state of the art classification, applications, and solution methods”. In: *Operational Research* 22.1 (2022), pages 1–47.

[143] Raad Salman, Fredrik Ekstedt, and Peter Damaschke. “Branch-and-bound for the precedence constrained generalized traveling salesman problem”. In: *Operations Research Letters* 48.2 (2020), pages 163–166.

[144] Tjendera Santoso, Shabbir Ahmed, Marc Goetschalckx, and Alexander Shapiro. “A stochastic programming approach for supply chain network design under uncertainty”. In: *European Journal of Operational Research* 167.1 (2005), pages 96–115.

[145] Joao Sarubbi, Gilberto Miranda, Henrique Pacca Luna, and Geraldo Mateus. “A cut-and-branch algorithm for the multicommodity traveling salesman problem”. In: *2008 IEEE International Conference on Service Operations and Logistics, and Informatics*. Volume 2. IEEE. 2008, pages 1806–1811.

[146] André Scholz. “An Exact Solto the Singleution Approach: Picker Routing Problem in Warehouses with an Arbitrary Block Layout”. In: *Working Paper Series* (2016).

[147] André Scholz, Sebastian Henn, Meike Stuhlmann, and Gerhard Wäscher. “A new mathematical programming formulation for the single-picker routing problem”. In: *European Journal of Operational Research* 253.1 (2016), pages 68–84.

[148] André Scholz and Gerhard Wäscher. “Order batching and picker routing in manual order picking systems: the benefits of integrated routing”. In: *Central European Journal of Operations Research* 25.2 (2017), pages 491–520.

[149] A Immanuel Selvakumar and K Thanushkodi. “A new particle swarm optimization solution to nonconvex economic dispatch problems”. In: *IEEE transactions on power systems* 22.1 (2007), pages 42–51.

[150] Lahari Sengupta, Radu Mariescu-Istodor, and Pasi Fränti. “Which local search operator works best for the open-loop TSP?” In: *Applied Sciences* 9.19 (2019), page 3985.

[151] Xiaohu H Shi, Yanchun Chun Liang, Heow Pueh Lee, C Lu, and QX Wang. “Particle swarm optimization-based algorithms for TSP and generalized TSP”. In: *Information processing letters* 103.5 (2007), pages 169–176.

[152] John Silberholz and Bruce Golden. “The generalized traveling salesman problem: A new genetic algorithm approach”. In: *Extending the horizons: advances in computing, optimization, and decision technologies*. Springer, 2007, pages 165–181.

[153] Stephen L Smith and Frank Imeson. “GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem”. In: *Computers & Operations Research* 87 (2017), pages 1–19.

[154] Lawrence V Snyder and Mark S Daskin. "A random-key genetic algorithm for the generalized traveling salesman problem". In: *European journal of operational research* 174.1 (2006), pages 38–53.

[155] Kenneth Sörensen and Marc Sevaux. "A practical approach for robust and flexible vehicle routing using metaheuristics and Monte Carlo sampling". In: *Journal of Mathematical Modelling and Algorithms* 8.4 (2009), pages 387–407.

[156] Curtis L Stowers and Udatta S Palekar. "Lot sizing problems with strong set-up interactions". In: *IIE transactions* 29.2 (1997), pages 167–179.

[157] Yixuan Su, Meixia Li, Xi Zhu, and Chunfa Li. "Steiner TSP based on aisle as a unit for order picking". In: *Computers & Industrial Engineering* 168 (2022), page 108026.

[158] Nicolas Teypaz, Susann Schrenk, and Van-Dat Cung. "A decomposition scheme for large-scale Service Network Design with asset management". In: *Transportation Research Part E: Logistics and Transportation Review* 46.1 (2010), pages 156–170.

[159] Christophe Theys, Olli Bräysy, Wout Dullaert, and Birger Raa. "Towards a metaheuristic for routing order pickers in a warehouse". In: *Evolutionary methods for design, optimization and control* (2007), pages 385–390.

[160] Christophe Theys, Olli Bräysy, Wout Dullaert, and Birger Raa. "Using a TSP heuristic for routing order pickers in warehouses". In: *European Journal of Operational Research* 200.3 (2010), pages 755–763.

[161] James A Tompkins, John A White, Yavuz A Bozer, and Jose Mario Azaña Tanchoco. *Facilities planning*. John Wiley & Sons, 2010.

[162] Rafael D Tordecilla, Angel A Juan, Jairo R Montoya-Torres, Carlos L Quintero-Araujo, and Javier Panadero. "Simulation-optimization methods for designing and assessing resilient supply chain networks under uncertainty scenarios: A review". In: *Simulation modelling practice and theory* 106 (2021), page 102166.

[163] Alessio Trivella, Francesco Corman, David F Koza, and David Pisinger. "The multi-commodity network flow problem with soft transit time constraints: Application to liner shipping". In: *Transportation Research Part E: Logistics and Transportation Review* 150 (2021), page 102342.

[164] C-Y Tsai, James JH Liou, and T-M Huang. “Using a multiple-GA method to solve the batch picking problem: considering travel distance and order due time”. In: *International Journal of Production Research* 46.22 (2008), pages 6533–6555.

[165] Edward Tsang and Chris Voudouris. “Fast Local Search and Guided Local Search and Their Application to British Telecom’s”. In: *TC* 10 (), page 1.

[166] Jimi Tuononen. “Analysis of rebuild local search operator for TSP”. Master’s thesis. Itä-Suomen yliopisto, 2022.

[167] Cristiano Arbex Valle, John E Beasley, and Alexandre Salles da Cunha. “Modelling and solving the joint order batching and picker routing problem in inventories”. In: *International symposium on combinatorial optimization*. Springer. 2016, pages 81–97.

[168] Cristiano Arbex Valle, John E Beasley, and Alexandre Salles da Cunha. “Optimally solving the joint order batching and picker routing problem”. In: *European Journal of Operational Research* 262.3 (2017), pages 817–834.

[169] Sarah Vanheusden, Teun van Gils, Katrien Ramaekers, Trijntje Cornelissens, and An Caris. “Practical factors in order picking planning: state-of-the-art classification and review”. In: *International Journal of Production Research* (2022), pages 1–25.

[170] TS Vaughan. “The effect of warehouse cross aisles on order picking efficiency”. In: *International Journal of Production Research* 37.4 (1999), pages 881–897.

[171] Bram Verweij, Shabbir Ahmed, Anton J Kleywegt, George Nemhauser, and Alexander Shapiro. “The sample average approximation method applied to stochastic routing problems: a computational study”. In: *Computational optimization and applications* 24.2 (2003), pages 289–333.

[172] Ulrike Von Luxburg and Bernhard Schölkopf. “Statistical learning theory: Models, concepts, and results”. In: *Handbook of the History of Logic*. Volume 10. Elsevier, 2011, pages 651–706.

[173] Christos Voudouris and Edward Tsang. “Guided local search and its application to the traveling salesman problem”. In: *European journal of operational research* 113.2 (1999), pages 469–499.

[174] Christos Voudouris, Edward PK Tsang, and Abdullah Alsheddy. “Guided local search”. In: *Handbook of metaheuristics*. Springer, 2010, pages 321–361.

[175] Zujian Wang, Mingyao Qi, Chun Cheng, and Canrong Zhang. "A hybrid algorithm for large-scale service network design considering a heterogeneous fleet". In: *European Journal of Operational Research* 276.2 (2019), pages 483–494.

[176] Felix Weidinger, Nils Boysen, and Michael Schneider. "Picker routing in the mixed-shelves warehouses of e-commerce retailers". In: *European Journal of Operational Research* 274.2 (2019), pages 501–515.

[177] Andres Weintraub, Julio Aboud, Claudio Fernandez, Gilbert Laporte, and Enrique Ramirez. "An emergency vehicle dispatching system for an electric utility in Chile". In: *Journal of the Operational Research Society* 50.7 (1999), pages 690–696.

[178] Mackenzie G Whitman, Kash Barker, Jonas Johansson, and Mohamad Darayi. "Component importance for multi-commodity networks: Application in the Swedish railway". In: *Computers & Industrial Engineering* 112 (2017), pages 274–288.

[179] Richard D Wollmer. *The Dantzig-Wolfe decomposition principle and minimum cost multicommodity network flows*. Technical report. RAND CORP SANTA MONICA CA, 1969.

[180] Jinhui Yang, Xiaohu Shi, Maurizio Marchese, and Yanchun Liang. "An ant colony optimization method for generalized TSP problem". In: *Progress in Natural Science* 18.11 (2008), pages 1417–1422.

[181] Xi Zhao and Xiao-Ping Zhu. "Innovative genetic algorithm for solving GTSP". In: *2010 Second International Conference on Modeling, Simulation and Visualization Methods*. IEEE. 2010, pages 239–241.

[182] Li Zhou, Zhaochan Li, Ning Shi, Shaohua Liu, and Ke Xiong. "Performance Analysis of Three Intelligent Algorithms on Route Selection of Fishbone Layout". In: *Sustainability* 11.4 (2019), page 1148.

[183] Ivan Žulj, Christoph H Glock, Eric H Grosse, and Michael Schneider. "Picker routing and storage-assignment strategies for precedence-constrained order picking". In: *Computers & Industrial Engineering* 123 (2018), pages 338–347.

Published work

- Rajabighamchi, F., van Hoesel, S., & Defryn, C. (2023). Graph reduction for the planar Travelling Salesman Problem: An application in order picking. Maastricht University, Graduate School of Business and Economics. GSBE Research Memoranda No. 004 https://doi.org/10.26481/umagsb.2023004
- Rajabighamchi, F., van Hoesel, S., & Defryn, C. (2023). The order picking problem under a scattered storage policy. Maastricht University, Graduate School of Business and Economics. GSBE Research Memoranda No. 006 https://doi.org/10.26481/umagsb.2023006
- Rajabighamchi, F., van Hoesel, S., & Defryn, C. (2023). Bus service for cargo: The periodic service network design problem with regular and express deliveries under demand uncertainty. Maastricht University, Graduate School of Business and Economics. GSBE Research Memoranda No. 005 https://doi.org/10.26481/umagsb.2023005

About the author

Farzaneh Rajabighamchi was born on December 27, 1991 in Urmia, Iran. She completed high school in 2010 and joined Isfahan University of Technology for Bachelor in Industrial Engineering. She obtained BSc degree in 2014 with specialization in Operations Research and Inventory control. Subsequently she attended MSc in Railway Engineering from the Iran University of Science and Technology in Tehran, Iran. In 2017, she obtained a Master degree of Transportation Engineering . She joined Maastricht University as a PhD candidate under the supervision of Prof. Dr. Stan van Hoesel, and Dr. Christoff Defryn and at the Logistics Group at UHasselt, under the supervision of Prof. dr. Inneke Van Nieuwenhuyse. The outcomes of her research are presented in this thesis. Some parts of this thesis have been presented at various European conferences. Her research interests are Operations Research, Optimization, Mathematical Modelling, Transportation and Supply Chain Management. In January 2023, Farzaneh started working as a lecturer at Maastricht University.