Material handling optimization in warehousing operations

Thèse

Thomas Chabot

Doctorat en sciences de l'administration - opérations et systèmes de décision
Philosophiæ doctor (Ph. D.)

Québec, Canada

Material Handling Optimization in Warehousing Operations

Thèse

Thomas Chabot

Sous la direction de:

Jacques Renaud, directeur de recherche
Leandro Callegari Coelho, codirecteur de recherche

Résumé

Les activités de distribution et d'entreposage sont des piliers importants de la chaîne d'approvisionnement. Ils assurent la stabilité du flux de matières et la synchronisation de toutes les parties prenantes du réseau. Un centre de distribution (CD) agit comme un point de découplage entre l'approvisionnement, la production et les ventes. La distribution comprend un large éventail d'activités visant à assurer la satisfaction de la demande. Ces activités passent de la réception au stockage des produits finis ou semi-finis, à la préparation des commandes et à la livraison. Les opérations d'un CD sont maintenant perçues comme des facteurs critiques d'amélioration. Elles sont responsables de la satisfaction d'un marché en évolution, exigeant des délais de livraison toujours plus rapides et plus fiables, des commandes exactes et des produits hautement personnalisés. C'est pourquoi la recherche en gestion des opérations met beaucoup d'efforts sur le problème de gestion des CDs. Depuis plusieurs années, nous avons connu de fortes avancées en matière d'entreposage et de préparation de commandes. L'activité de préparation de commandes est le processus consistant à récupérer les articles à leur emplacement de stockage afin d'assembler des commandes. Ce problème a souvent été résolu comme une variante du problème du voyageur de commerce, où l'opérateur se déplace à travers les allées de l'entrepôt. Cependant, les entrepôts modernes comportent de plus en plus de familles de produits ayant des caractéristiques très particulières rendant les méthodes conventionnelles moins adéquates.

Le premier volet de cette thèse par articles présente deux importants et complexes problèmes de manutention des produits lors de la préparation des commandes. Le problème de préparation des commandes a été largement étudié dans la littérature au cours des dernières décennies. Notre recherche élargit le spectre de ce problème en incluant un ensemble de caractéristiques associées aux installations physiques de la zone de prélèvement, comme les allées étroites, et aux caractéristiques des produits (poids, volume, catégorie, fragilité, etc.). Une perspective plus appliquée à la réalité des opérations est utilisée dans notre développement d'algorithmes.

Les déplacements liés à la préparation des commandes sont fortement influencés par le positionnement des produits. La position des produits dans la zone de prélèvement est déterminée par une stratégie d'affectation de stockage (*storage assignment strategy*). Beaucoup de ces stratégies utilisent de l'information sur les ventes des produits afin de faciliter l'accès aux plus

populaires. Dans l'environnement concurrentiel d'aujourd'hui, la durée de vie rentable d'un produit peut être relativement courte. Des promotions peuvent également être faites pour pousser différents produits sur le marché. Le positionnement fourni par la stratégie d'hier ne sera probablement plus optimal aujourd'hui. Il existe plusieurs études mesurant l'impact d'une bonne réaffectation de produits sur les opérations de prélèvement. Cependant, ils étudient la différence des performances avec les positionnements passés et actuels. La littérature démontre clairement que cela apporte des avantages en termes d'efficacité. Toutefois, les déplacements nécessaires pour passer d'une position à une autre peuvent constituer une activité très exigeante. Ceci constitue le second volet de cette thèse qui présente des avancées intéressantes sur le problème de repositionnement des produits dans la zone de prélèvement. Nous présentons le problème de repositionnement des produits sous une forme encore peu étudiée aux meilleurs de nos connaissances : le problème de repositionnement. Plus précisément, nous étudions la charge de travail requise pour passer d'une configuration à l'autre.

Cette thèse est structuré comme suit. L'introduction présente les caractéristiques et les missions d'un système de distribution. Le chapitre 1 fournit un survol de la littérature sur les principales fonctions d'un centre de distribution et met l'accent sur la préparation des commandes et les décisions qui affectent cette opération. Le chapitre 2 est consacré à l'étude d'un problème de préparation de commandes en allées étroites avec des équipements de manutention contraignants. Dans le chapitre 3, nous étudions un problème de préparation des commandes où les caractéristiques des produits limitent fortement les routes de prélèvement. Le chapitre 4 présente une variante du problème de repositionnement (*reassignment*) avec une formulation originale pour le résoudre. La conclusion suit et résume les principales contributions de cette thèse.

Mots clés : Préparation des commandes, entreposage, problèmes de routage, algorithmes exacts et heuristiques, réaffectation des produits, manutention.

Abstract

Distribution and warehousing activities are important pillars to an effective supply chain. They ensure the regulation of the operational flow and the synchronization of all actors in the network. Hence, distribution centers (DCs) act as crossover points between the supply, the production and the demand. The distribution includes a wide range of activities to ensure the integrity of the demand satisfaction.

These activities range from the reception and storage of finished or semi-finished products to the preparation of orders and delivery. Distribution has been long seen as an operation with no or low added value; this has changed, and nowadays it is perceived as one of the critical areas for improvement. These activities are responsible for the satisfaction of an evolving market, requiring ever faster and more reliable delivery times, exact orders and highly customized products. This leads to an increased research interest on operations management focused on warehousing. For several years, we have witnessed strong advances in warehousing and order picking operations. The order picking activity is the process of retrieving items within the storage locations for the purpose of fulfilling orders. This problem has long been solved as a variant of the travelling salesman problem, where the order picker moves through aisles. However, modern warehouses with more and more product families may have special characteristics that make conventional methods irrelevant or inefficient.

The first part of this thesis presents two practical and challenging material handling problems for the order picking within DCs. Since there are many research axes in the field of warehousing operations, we concentrated our efforts on the order picking problem and the repositioning of the products within the picking area. The order picking problem has been intensively studied in the literature. Our research widens the spectrum of this problem by including a set of characteristics associated with the physical facilities of the picking area and characteristics of the product, such as its weight, volume, category, fragility, etc. This means that a more applied perspective on the reality of operations is used in our algorithms development.

The order picking workload is strongly influenced by the positioning of the products. The position of products within the picking area is determined by a storage assignment strategy. Many of these strategies use product sales information in order to facilitate access to the most popular items. In today's competitive environment, the profitable lifetime of a product can be

relatively short. The positioning provided by yesterday's assignment is likely not the optimal one in the near future. There are several studies measuring the impact of a good reassignment of products on the picking operations. However, they study the difference between the two states of systems on the picking time. It is clear that this brings benefits. However, moving from one position to another is a very workload demanding activity. This constitutes the second part of this thesis which presents interesting advances on the repositioning of products within the picking area. We introduce the repositioning problem as an innovative way of improving performance, in what we call *the reassignment problem*. More specifically, we study the workload required to move from one setup to the next.

This thesis is structured as follows. The introduction presents the characteristics and missions of a distribution system. Chapter 1 presents an overview of the literature on the main functions of a DC and emphasizes on order picking and decisions affecting this operation. Chapter 2 is devoted to the study of a picking problem with narrow aisles facilities and binding material handling equipment. In Chapter 3, we study the picking problem with a set of product features that strongly constrain the picking sequence. Chapter 4 presents a variant of the reassignment problem with a strong and new formulation to solve it. The conclusion follows and summarizes the main contributions of this thesis.

Key words: Order-picking, warehousing, routing problems, exact and heuristic algorithms, products reassignment, material handling.

Contents

List of Tables

List of Figures

Acknowledgments

First and foremost, I wish to thank my advisors, Professor Jacques Renaud and Professor Leandro C. Coelho. Jacques supports me since my very first year at the master degree. He taught me how to become a better student, researcher and professional. But more than anything, he taught me to have pleasure in what I do, even if it is not always easy.

Since his arrival at Université Laval, Leandro inspires passion for what he does and his appetite for applied research has allowed me to explore so much new knowledge. Through this long journey, I had ups and downs. Leandro was there when I needed it most and always allowed me to rise stronger.

Dear directors, I will never be able to thank you enough for your constant kindness, availability and your daily support. This work would not have been possible without the two of you.

I wish to thank my committee members : Professors Fayez Fouad Boctor, Monia Rekik and Walter Rei for their time and support.

I would like to thank administrative and technical staff members at the Faculty of Business Administration, the Ph.D. program committee, and the CIRRELT for their everyday help. I also thank the professors, my friends, and colleagues from the CIRRELT.

Last but not least, I would like to offer my heartfelt thanks to my friends and family who have been there with me. My dear parents, you have always encouraged me to go further and surpass myself in everything. You have always made sure that I am happy and fulfilled, which allowed me to go through these years with pleasure.

Preface

This thesis presents my work as a Ph.D. student developed at the Centre Interuniversitaire de recherche sur les Réseaux d'Entreprise, la Logistique et le Transport (CIRRELT) at the Faculty of Business Administration of Laval University. The thesis consists of three papers, each of which is written in collaboration with other researchers, mainly my directors Jacques Renaud and Leandro Callegari Coelho. Two of these papers are already published and the other is ready to be submitted. In all three papers, I remain the first author and have played the major role of setting up and conducting the research, modeling and implementation of the problem and algorithms, analyzing the results, preparing and writing the papers.

The first paper entitled *Mathematical model, heuristics and exact method for order picking in narrow aisles* is written in collaboration with Leandro C. Coelho, Jacques Renaud and Jean-François Côté. The paper is accepted by the Journal of the Operational Research Society in September 2017 and was published online in December 2017.

The second paper entitled *Order picking problems under weight, fragility and category constraints* is written in collaboration with Rahma Lahyani, Leandro C. Coelho and Jacques Renaud. The paper is accepted by the International Journal of Production Research in September 2015 and was published online in October 2016.

The third paper entitled *Mathematical models for the warehouse reassignment problem* is written in collaboration with Leandro C. Coelho and Jacques Renaud. The paper has been submitted for publication in Computers & Operations Research in February 2018.

These three papers constitute Chapter 2, 3 and 4 of this thesis. In addition to this thesis, I have published two others paper. The first one, entitled *Biomedical sample transportation in the province of Quebec: a case study*, has been written in collaboration with Ana Maria Anaya-Arenas, Jacques Renaud and Angel Ruiz. This paper has been published in the International Journal of Production Research in 2015. The second paper entitled, *Service level, cost and environmental optimization of collaborative transportation*, has been written in collaboration with Florence Bouchard, Arianne Legault-Michaud, Jacques Renaud and Leandro C. Coelho. It was published in Transportation Research Part E: Logistics and Transportation Review in 2017.

Introduction

Warehousing and distribution activities are critical and among the most important activities in the supply chain. Several products such as raw materials, products-in-progress and finished products need to move from a location to another (origin, manufacturers, end users, etc.). Sometimes these products will have to be stored, serving as a buffer stock during a certain period. Warehousing activities influence the speed of the supply network and therefore have significant logistics and financial impacts. Manufacturing strategies cannot be efficiently deployed without a systematic control of distribution activities. Distribution activities exist throughout the logistics network: in the supply of the raw materials, between the production units, to the assembly until the final distribution to the customers and so on. The distribution center (DC) system also significantly affects the product quality, the consumers service level and the logistic costs.

The global mission of a DC is the same as the traditional definition of logistics - efficiently delivering products to the right place at the right time, in good quantity and without damage [Slack, 2015]. However, operating a DC often involves large investments and operating costs (i.e., cost of land, facility equipment, labor). So, why do DCs exist and why they are now essential? According to Lambert et al. [1998] they contribute to a multitude of the company's missions, such as:

— achieving transportation economies (e.g., combine shipment, full-container load),
— achieving production economies,
— taking advantage of purchase discounts,
— supporting the firm's customer service policies,
— meeting changing market and uncertainties (seasonality and demand fluctuations),
— overcoming the time and space differences that exist between producers and customers,
— accomplishing least cost logistics with a desired level of customer service,
— supporting just-in-time programs,
— providing customers with a mix of products instead of a single product on each order,
— providing temporary storage for reverse logistics, and
— providing a buffer location for cross-docking.

There have been many innovations in warehouse management in recent decades. DCs now provide value-added activities or services in order to be more functional. One finds activities and services such as product consolidating, cross-docking, quality checking, final assembling, packaging, reverse logistics, information services, etc. [Le-Duc, 2005]. In this perspective, it is not surprising to see the DCs becoming bigger and also much more flexible. Hence, they need various technologies to become faster, while responding to the demand and managing a multiplication of products. As Le-Duc [2005] presents, the industry tends to consolidate their warehousing activity in order to reduce the network management effort. It also allows to reduce the safety stock (less incertitude with fewer storage locations) and to gain economies of scale. In fact, the DC management became such a complex operation that several companies prefer to outsource this activity in order to focus in their core business.

Operating a DC implies several operations. Among them, we have the order picking as the process of retrieving items within the storage locations for the purpose of fulfilling a customer's order. It is identified as a very labor intensive operation in manual systems, and a very capital intensive operation in automated systems [Goetschalckx and Ratliff, 1988]. Order picking may consume as much as 60% of all labor activities and cost in the DC [Drury, 1988, Frazelle, 2002, Tompkins et al., 2010]. This explains why warehousing professionals and researchers are putting so much effort into the optimization of this process. It is also the main theme of this thesis. The literature in this field is very rich and will be properly explored and categorized in Chapter 1.

This research is positioned on high volume and technologically advanced DCs. This type of DC is flexible and adapted to frequent changes in demand pattern and product types. This high level of flexibility requires the development of specific planning methods to address this dynamism. Demand analysis and forecasting tools have been developed to become more accurate. In Chapter 1, we study the spectrum of activities that have a direct impact on the performance of a DC, in particular activities impacting and related to order picking operations. The chapter explains how each main function relates to the order preparation, from the input to the output of the DC, have to be managed together in order to have the best overall system. These include receiving activities, storage strategies, picking routing strategies, layout design and the choice of equipment, among others.

The reminder of this thesis is organized as follows. In Chapter 2 we address some physical characteristics of a DC containing what is called *narrow aisles*. This type of structure imposes particular operational constraints, which make classical and naive picking routing methods inefficient. Our hypothesis is that we can significantly improve a classical routing exact method by incorporating practical features of the problem. We show that one needs to properly consider and model all distances between pairs of product locations realistically, considering the structure of the DC. We solve the problem by exact and heuristic methods and solve instances containing up to 480 locations per aisle.

In Chapter 3 we extend our research to consider general DC structures, but we study many properties arising from the characteristics of the products. These are, for example, products that are fragile and could be crushed if heavy ones are placed on top of them, or products that could contaminate others. These characteristics impose many constraints on the planning of a picking tour, and we study their impact from an algorithmic performance. Again, we propose different mathematical models and several heuristic methods. We test all methods on large instances containing up to 100 picks.

In Chapter 4 we propose a way to change the assignment of products to their locations. As we will show, many studies have demonstrated the value of updating the product assignments on the picking performance, but they do not show how to change from one assignment to another. Here, we define what we call the *reassignment problem*. We formally define this problem in a different, more efficient and realistic way from what the literature currently proposes.

Whether it is about the physical characteristics of the installations, about the products or about the order picker safety, one of the main contributions of this thesis is to clearly formulate practical warehousing problems. We present different alternatives, from heuristics to exact methods to solve these problems, but also which can be easily adapted to similar problems with different constraints. The literature tends to present new order picking problems with some real-life constraints. However, we show that the literature presents research for solving hard problems with very precise methodology, lacking generic resolution methods that can be reused for similar problems. This is what we try to avoid. Another important contribution of this thesis is the study of a new formulation for a still little studied problem: the reassignment of products. There is actually very few research trying to compute exactly the workload needed to pass from a storage assignment to another. This problem is very important in real life since warehouses often need to update their assignment. In our research, we develop a new routing-based formulation indicating the order of all travel movements and product exchanges. We hence compare this new formulation with an existing formulation in order to validate the strength of our method. Our conclusions summarizes our main contributions.

Chapter 1

Literature Review

As we presented in the Introduction, warehouse management is an important area of research. There is a large number of factors that have forced a continuous optimization of distribution centers. As presented by Lu et al. [2016], one of them is reduced inventory combined with the reduction of response time which put great pressure on the logistics network [Agarwal et al., 2006, Naim and Gosling, 2011]. We see greater number of third parties logistic providers (3PLs) managing larger, more complex warehouses with multiple customers with different requirements [Selviaridis and Spring, 2007]. Finally, one of the most important factors is the constant rise of e-commerce, in which a large number of small orders are managed directly by warehouses [Gong and De Koster, 2008, Davarzani and Norrman, 2015, De Koster et al., 2017]. These factors push researchers to develop even better techniques in response to growing needs. At the DC level, this is achieved by optimizing order preparation.

As presented in Park [2012], an order is a list of one or more lines, each line representing a needed stock-keeping-unit (SKU), going to a specific customer/destination. A line is a separate item of supply on a transaction document. A SKU represents one unique inventory item in the smallest physical unit handled in a warehouse. Thus, an item line and the requested quantity should be expressed in terms of SKUs. Customer orders can be converted into one or many pick lists for the picking operation within the warehouse. A pick list is a list of one or more pick lines that typically consists of a SKU, its quantity to be picked, and the storage location.

Order picking is a crucial operation to optimize if we want to minimize costs and preparation time. Since orders are increasingly coming directly to the warehouse via the online shopping, efficiency must always be greater. Considering a high number of orders with few products from the online shopping habits, it is easy to imagine how complex and dynamic the order picking becomes. Even if the time between an order release and the time to reach its final customer destination is always smaller, there is still ample opportunities for errors in accuracy, completeness and time lost [De Koster et al., 2007]. This is true even when an order has just two products to pick, so when an order contains hundreds of products, this error probability

becomes important.

There can be more than one product family in a DC and it may imply several technical and practical constraints. Some companies, like supermarkets, sell different product families. They sell food and products such soaps, cleansers, frozen, oil, etc. One does not want to mix some products in the same bags. The problem is exactly the same in the DC. The picking operation should consider this type of practical constraints in order to avoid, for example, contamination of products. To facilitate these operations, products must be positioned strategically within aisles and picking wisely.

The literature on warehousing operations is so vast that it is almost impossible to cover. Davarzani and Norrman [2015] perform a literature review to identify distinct research areas. Some of these are illustrated in Figure 1.1. This framework is based on material handling operations such as reception, storage, picking, packing, and shipping products and supportive entities like strategy, infrastructure design, human resource management, technology and performance evaluation. Most of the literature focus on the material handling part.

<table>
<tr><td colspan="4">Operation strategy</td></tr>
<tr><td colspan="4">Infrastructure design</td></tr>
<tr><td colspan="4">Human resource mangement</td></tr>
<tr><td colspan="4">Technology</td></tr>
<tr><td colspan="4">Performance evaluation</td></tr>
<tr><td>Receiving</td><td>Storage</td><td>Order picking</td><td>Shipping</td></tr>
</table>

Figure 1.1 – Warehousing reasearch area, inspired by [Davarzani and Norrman, 2015]

The classification of only material handling problems is also very vast, from strategic decisions to operational ones. Van Gils et al. [2017b], inspired by De Koster et al. [2007], offer a recent state-of-the-art review and classification of the order picking problems. Some examples of these problems are presented in Figure 1.2 and summarize the material handling problems classification for a reasonable part of the literature within this spectrum. This chapter presents an overview of some of these decision problems. Particularly the layout of the storage area for the strategic point of view, the storage assignment for the tactical one and ending with batching and order picking routing problems for the operational decisions.

This chapter begins with an explanation of the flow of material within the warehouse. Most of these constitutes what is illustrated in the last row of Figure 1.1. Afterwards, it presents several order picking methods employing humans or machines to operate and the main objectives and components of the picking operation. It reviews important decisions on warehouse design that have an impact on the order picking performance. Finally, the chapter concludes with an overview of the products reassignment problem literature. See De Koster et al. [2007], Le-Duc [2005] and Van Gils et al. [2017b] for a more exhaustive overview of the different warehouse functions and classification.

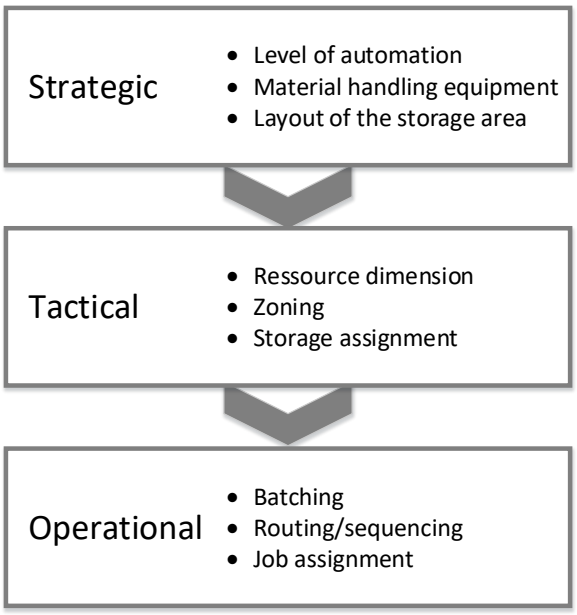


Figure 1.2 – Order picking problems classification, inspired by [Van Gils et al., 2017b]

1.1 Distribution center functions and operations

Products have to pass by a series of operations before getting ready to be shipped to the final customer. Figure 1.3 shows the typical functional areas and flows within warehouses as presented in Tompkins et al. [2010]. The main warehouse activities include: receiving, transfer to the storage reserve and/or picking area, order picking, accumulation/sorting and shipping. We can also add the cross-docking within accumulation and sortation area.

The receiving activity includes the unloading of products from the transport carriers, updating the inventory record, labelling and inspection to determine whether there is any quantity or quality problems. It is the activity that ensures products to enter appropriately in the warehouse and to be available for positioning and picking. In some large volume industries, such as food retail, this activity is important as there are many different deliveries to the warehouse each day.

The transfer to the reserve area (in Figure 1.3: Direct put away to reserve/primary) involves the movement of incoming products to storage locations. It may also include repackaging (in order to fit with the own company packages), and physical movements from the receiving docks to different functional areas. Before transferring the product to the picking location or the reserve, a storage assignment policy should have been decided. This consists of determining an ideal location for each product within the reserve or the order picking area. This has a direct impact on the performance (time, distance, errors) of the picking operation [Petersen, 1997] and on the space utilization of the DC. In this step, it is usual to update location information

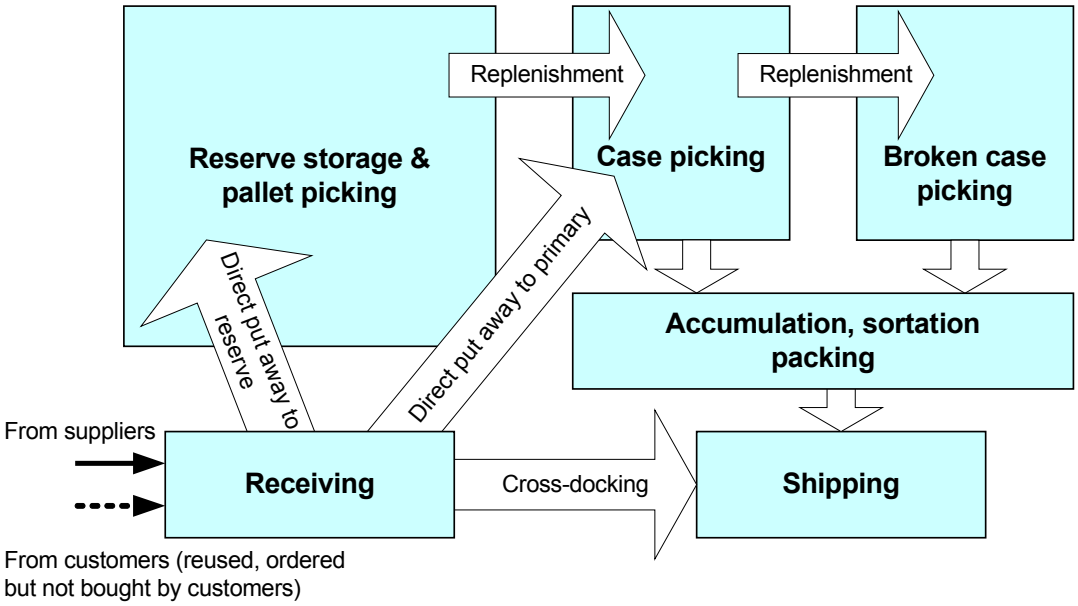


Figure 1.3 – Typical warehouse functions and flows [Tompkins et al., 2010]

of products. Section 1.4 details the storage assignment problem.

Order picking involves the determination of the best way to pick all the items specified on one or many picking lists. This is the main subject of this chapter and Sections 1.2 and 1.3 are devoted to more extensive explanations of this step and its research literature. Section 1.4 explains a series of warehouse problems that have an important impact on the order picking process. When a reserve area is used, replenishment activity of the order picking area (fast pick area) will take place. The order picking activities will reduce inventory in this area and it will have to be replenished continuously. Hackman et al. [1990] develop a model to decide which products should be assigned to the picking area and how much space must be allocated to each of the products given a fixed storage capacity of the forward area. Frazelle et al. [1994] extend the problem and the solution method of Hackman and Platzman [1990] by treating the size of the forward area as a decision variable. Yu and De Koster [2010] study this problem in a dynamic environment where products are continually relocated.

The consolidation of orders is a necessary activity in which orders will be picked in batches (order batching). An order batching procedure is often necessary when the DC receives a lot of small orders. It allows to reduce the number of picking tours and the movement within aisles. This activity involves sorting the orders after the picking. We discuss this in more details Section 1.4.3.

After the picking, orders often have to be packed and stacked on the right unit load (e.g., a pallet). The products carried by the order pickers are consolidated and packed according to the original orders. At this step, the link is made between the warehouse and external

distribution systems. The classic problems of this step are the selection of handling units, the minimization of damage, the handling minimization between picking and loading of carriers and management of queues at loading docks. In this phase, orders are sorted according to customers and/or delivery vehicles, which in turn are assigned to different loading docks. Everything must be in the right place at the right time. Vehicle loading may be a complex problem when using a trailer with a single rear door (no lateral door). In this case, orders must be positioned in the trailer according to the visits sequence of the delivery route (first-in last-out). We can also see cross-docking operations performed when the received products are transferred directly to the shipping docks (no order picking is necessary).

1.2 Order picking

As previously mentioned, order picking involves the process of batching customer orders (when applicable), releasing them (picking list queue) and picking SKUs from storage locations. There are different order picking strategies that can be used by a warehouse. In some warehouses, there can be more than one of these (different equipment, different product families, etc.). Before discussing the literature on the order picking optimization, let explains several systems as presented in De Koster et al. [2007]. Figure 1.4 separates the order picking methods in two main families: manual and automated systems.

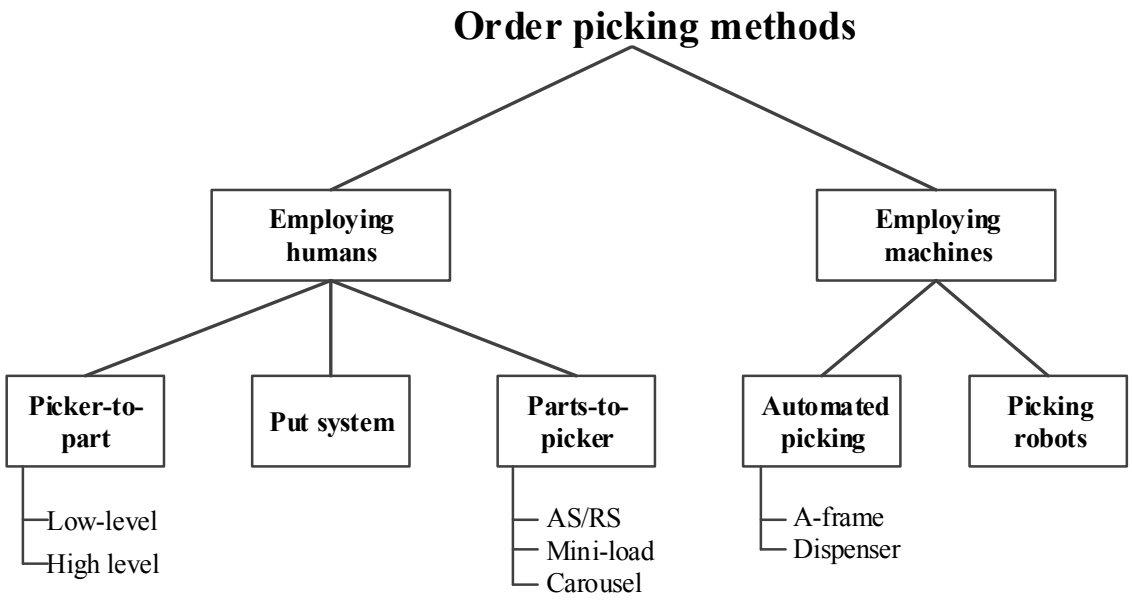


Figure 1.4 – Classification of order-picking systems (based on De Koster [2012], De Koster et al. [2007])

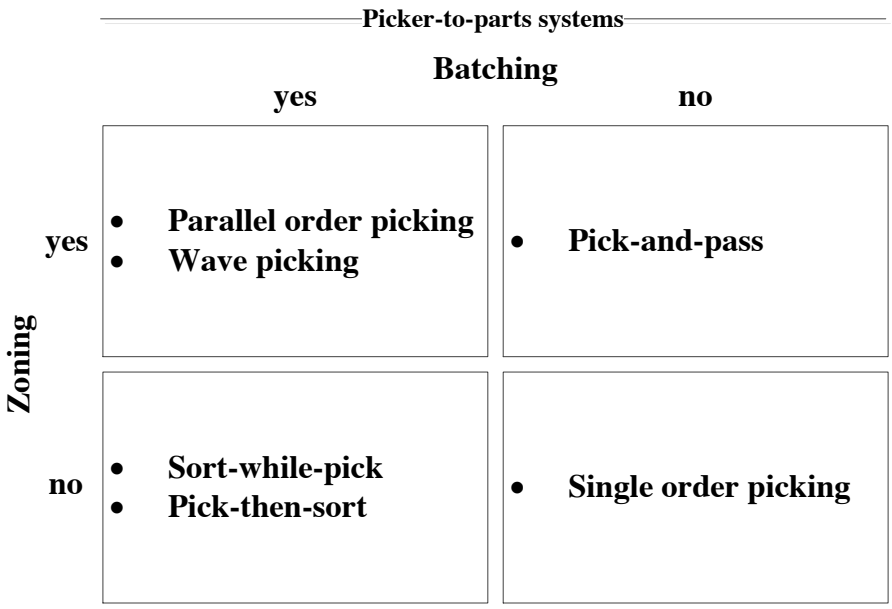


Figure 1.5 – Classification of picker-to-parts systems based on batching and zoning

The majority of warehouses use the manual (human) system. Among these, the most common is the *picker-to-parts system*. In this system, order picker moves with its picking vehicle within the aisles. There are two main types of picker-to-parts systems. The first one is the low-level picking where operators pick requested items from storage racks or bins (shelving storage) at the ground level. Because of the labour intensity, low-level systems are sometimes called manual-pick systems. Chapter 3 presents an example of low-level picking system. Some other order picking systems have high storage racks (more than one level). In this case, pickers travel to the pick locations on board of a lift truck or crane (man-aboard). The picker stops in front of the appropriate pick location and uses his vehicle to reach the target bin. Chapter 2 presents an application of high-level order picking problem.

Figure 1.5 shows variants of picker-to-part systems. The first basic variants is either the picking is done by group of articles (*batch picking*) or by picking by order (*discrete picking*). In the picking by group of articles, multiple articles from several orders (a batch) are picked simultaneously by a picker. Another variant of the picker-to-part system is the utilization of picking zones. The picking area is separated in multiple zones with their own picker and groups of products. We often see this kind of zone when the warehouse contains different families of products (liquid, frozen, heavy, etc.).

When no batching or zoning are used with a picker-to-parts system, we have a discrete order picking in which each order picker retrieve one complete order at a time. If orders are splitted, using batching, one can ensure the order integrity using a sort-while-pick or a pick-then-sort system. When using zoning of the picking area, the order integrity is ensured by a system

such as the pick-and-pass. A parallel order picking can be used when using both batching and zoning. In this, all order pickers start at the same time with their part of a set of orders. Each order picker picks the products of those orders from his zone. Accumulation is performed afterwards.

Parts-to-picker systems include automated storage and retrieval systems (AS/RS), using automated moving cranes that retrieve SKUs and bring them to a pick position. One disadvantage of this system is that in many cases a unit load is moved to the picker. As the picker may not need such a quantity, the automated system has to replace the remaining products. This kind of AS/RS can work under different operating modes: single, dual and multiple command cycles as presented in Sarker and Babu [1995]. The *single*-command cycle means that one load is moved from the input/output (I/O) location to a rack location and return empty or vice-versa. In the *dual*-command mode a load is moved from the I/O to the rack location and another load is retrieved from the rack to the I/O during the same cycle. In *multiple*-command cycles, the AS/RS can pick up and drop off several loads in one cycle (multiple compartments). Other systems use modular vertical lift modules, or carousels that also offer unit loads to the order picker, who is responsible for taking the right quantity. With new technological advances, we also see the emergence of ingenious systems. For example, the automated material handling solution such as the Kiva Mobile Robotic using a fleet of mobile robotic drive units to move shelves to the picker. To pick orders, operators stand at stations around the perimeter of the building while inventory is stored on mobile shelving racks, called pods. Large companies like Amazon are using this kind of new technology [Kirsner, 2012].

The complete *automated picking* and *picking with robots* system is much less used in practice and therefore much less present in the scientific literature. These methods are used in very technologically advanced warehouses and are very expensive. They are used in very special cases as with valuable, small and delicate items [De Koster et al., 2007].

In the order picking part of this thesis (Chapters 2 and 3), we concentrate on picker-to-parts picking systems employing humans with multiple picks per route.

1.3 Order picking objectives and components

The most common objective of order picking systems is to maximize the service level (often seen as minimizing the lead time) subject to resource constraints such as labor, machines, and capital [Goetschalckx and Ashayeri, 1989, Bartholdi and Hackman, 2011]. As mentioned before, the order picking process must ensure that orders will be prepared with accuracy and on time. An order must be prepared on time in order to leave sooner with the first available carrier. If it cannot be prepared on time, the order will have to wait until the next shipping period. Short picking times give the opportunity of handling late changes and offering short delivery windows. Minimizing the lead time is, therefore, a need for any order picking system.

Figure 1.6 shows the picking time components in a typical picker-to-parts warehouse. De Koster et al. [1999a] and Dekker et al. [2004] show that activities other than travel may substantially contribute to picking time. The travel time part remains the main component of the total picking operation time. According to Bartholdi and Hackman [2011] travel time is a waste. It costs labor hours but does not add any kind of value. It is, therefore, the first candidate for improvement. Figure 1.6 shows that the travel time represents 50% of the total time, and the search time (often considered in the travel time) up to 20%. With this, it is possible to say that 70% of the total-picking time is not directly related to material handling. For these reasons warehousing operation and order preparation studies are needed to reduce the travel time.

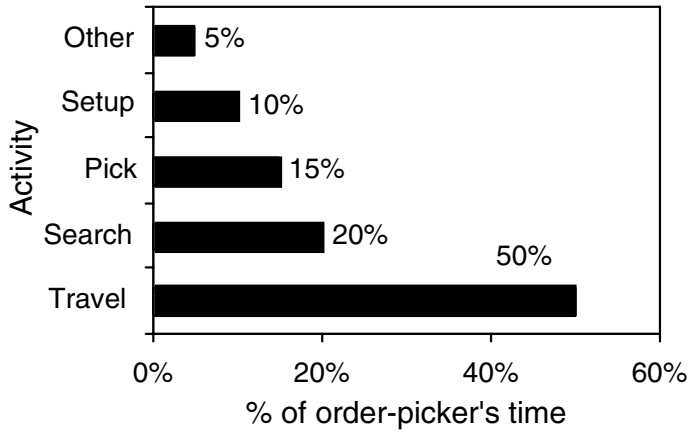


Figure 1.6 – Typical distribution of an order picker time (based on Tompkins et al. [2010])

For manual order picking systems, the travel time is an increasing function of the travel distance as presented in Jarvis and McDowell [1991], Hall [1993], Petersen [1997, 1999], Roodbergen and De Koster [2001a,b], Petersen and Aase [2004]. It is then equivalent to reduce the distance or time. It has been shown that minimizing the total travel time, the main objective minimized in most researches, is equivalent to minimizing the average time per route. Another important objective is minimizing the total cost, that may include both investment and operational costs. Other objectives could be the minimization of the time per order, the use of equipment and labor [De Koster et al., 2007]. Recently, some authors consider human factors in the optimization objective [Battini et al., 2017, Grosse et al., 2017].

1.4 Decisions affecting order picking operations

There are several design decisions that directly or indirectly affect the performance of the order picking system. These families of warehouse design problems are often seen at the tactical or operational levels [Van Den Berg, 1999, Rouwenhorst et al., 2000]. Common decisions are those related to layout design (dimensioning of storage systems), storage assignment, batching,

zoning, routing and order accumulation/sorting methods. The layout design and the storage assignment are more tactical decisions while the others are more operational. These decisions are what we call the design and planning of warehousing systems. They have been largely studied for warehousing systems in Ashayeri and Gelders [1985], Cormier and Gunn [1992], van den Berg [1999], Rouwenhorst et al. [2000], Roodbergen [2001], Gu et al. [2010b] and Roodbergen et al. [2015]. Combining the decision of tactical and operational planning horizons in a single model is still intractable [De Koster et al., 2007]. Thus, researchers limit their scope to one or few decision areas simultaneously. In practice, decisions are also made sequentially, but we see innovative research combining two or more control policies such as layout, storage policies and routing (tactical and operational) in the same methodology [Roodbergen et al., 2015]. Sprock et al. [2017] propose and formalize an up to date hierarchical design decision support methodology based on decomposing the design problem into a set of subproblems. In this section, we explain each of these decisions in more details for a manual order picking system.

1.4.1 Layout design

The layout design (physical arrangement of the warehouse, including the reserve and picking area) contains multiple decisions impacting the order picking. It involves locating the various departments illustrated in Figure 1.3 (receiving, picking, storage, shipping, etc.). The problem is to find a layout setup in which the material handling flow/costs are minimized. Meller and Gau [1996] and Tompkins et al. [2010] make a description and review of several layout design procedures. Figure 1.7 from De Koster et al. [2007] presents decisions within the order picking area. The designer has to determine the length and number of aisles, where to locate the I/O point (depot), the presence of cross-aisles, and if so, how many and where. He also chooses the type of racking, their number and their positioning. The objective is to find the best picking layout in order to minimize the order picking costs.

Roodbergen et al. [2008] and Gu et al. [2010a] present the steps of a typical design project for a picking area. First, one must determine the area and the choice of shelving and handling equipment. Then the physical structure of the aisles must be decided (wide, narrow, one-way) and finally the control policies managing the picking operations. Control policies are for example storage assignment procedures and routing strategies. Roodbergen [2001] proposed a non-linear objective function with the average travel time in terms of the number of picks per route and pick aisles. This helps determine the aisle configuration for random storage warehouses that minimizes the average tour length. Gu et al. [2010a] present key performance indicators (KPI) that are important for both warehouse design and operations. Evaluating performance in terms of cost, throughput and use of space provides feedback on the design quality. These generic performance evaluation and design steps are described in more detail in a review by Baker and Canessa [2009]. Roodbergen et al. [2015] present different design

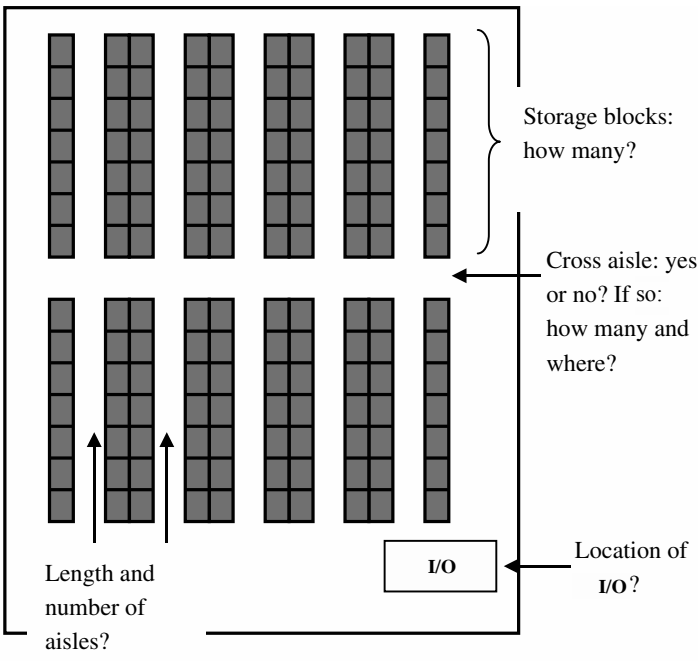


Figure 1.7 – Example of layout decisions for the order picking problem [De Koster et al., 2007]

policies for more details and apply them to a real case study to determine the best aisle configuration and control policy. They develop a simulation tool to estimate the average order picking distances including the innovative order picking layouts proposed by Gue and Meller [2009] and Gue et al. [2012]. These layouts have new type of cross-aisle and fishbone aisles which result in reduced travelling distance to retrieve SKUs, and consequently in improved efficiency in the storage area. The layout design problem for AS/RS has also received a lot of attention. For more details, readers are directed to Sarker and Babu [1995], Johnson and Brandeau [1996], van den Berg [1999], Le-Duc [2005], De Koster et al. [2007], Park [2012], Roy et al. [2017].

1.4.2 Storage assignment policies

After being received, products must be allocated to storage locations before the order picking process can take place. A *storage assignment policy* is a set of rules to determine the allocation of a SKU to a physical location. Heskett [1963] was one of the first to formalize a relation between products location and their characteristics. He established the relation between the number (or volume) of products to be allocated and their demand frequency. A popular method is to separate the picking zone in two areas: a forward area (also called fast-picking area) and a reserve area. The forward area is designed for fast picking operations while the reserve is used to replenish the first one. In general, the forward area is smaller in order to increase product density and to minimize the picking distance and does not contain all the products. The selection of products in the forward area and its configuration have a major

impact on the efficiency of picking operations [Walter et al., 2013]. As Gu et al. [2010a] showed, the problem of product selection (and how many) to be placed in the forward area is comparable to a backpacking problem and its resolution is then NP-hard.

There exists several classical storage assignment policies as presented in Hausman et al. [1976] and reviewed in Roodbergen [2012] and Kofler et al. [2014]:

— dedicated storage assignment,

— random storage assignment,

— full-turnover-based storage assignment and,

— class-based storage assignment.

These strategies can be adapted in the case of multiple blocks, where aisles are divided by cross-aisles (see Figure 1.7). In a dedicated storage policy, a specific location is reserved for each product. Products will always be placed in their respective locations. This minimizes the need to use an information system to locate products because the picker knows by experience each location. In the random assignment policy, products are assigned to a random location (with equal probability) in the warehouse. When a location has been emptied (by picking operations), any new incoming product can take its place. If there are multiple empty locations, they have equal probability to be selected [Petersen, 1997]. Under the closest open location policy, a new product is always placed at the nearest empty location. This random storage policy is regularly used in practice because it allows to use the warehouse evenly and avoid congestion. In return, it requires the use of a system for locating products in the warehouse in order to carry out the picking operation [De Koster et al., 1999a]. Using a random storage with the closest open location strategy typically leads to a warehouse where racks are full around the I/O and gradually emptier towards the back (if there is excess capacity).

The full-turnover-based storage policy needs more information about products. They are ranked from the most requested to the least. Locations are ranked by distance to the I/O point location. Then the products are assigned by respecting these two rankings, the most requested product in the best location. Periodic relocation should be made depending on promotions and product life cycles, but in general, this policy is dedicated. Unlike the random method, the use of the warehouse will be less uniform and may cause congestion in the high-sales zone.

A class-based storage policy combines the random method and the full-turnover-based method. It divides all the products into several groups or classes. Each class has its own storage zone. The assignment of locations is in general random within a class zone. This saves some internal congestion problems. Classes can be created in different ways. The most common method is to separate products according to their demand, i.e., a *Pareto ABC* classification. The Pareto law interpretation for our problem suggests that 20% of products tend to represent 80% of sales. These fast-moving products are referred as A-items. The following are the B-items and the least requested are the C-items. Each category (A, B and C) is assigned to a zone.

Then the sizing of each zone must be done accordingly. Note that the number of zones is not restricted.

Figure 1.8, from Roodbergen [2012], shows four examples of configurations of class-based storage policy with ABC classification. Black squares are locations occupied by A-items, dark gray are for B-items and light grey for C-items. It shows four ways to assign products to locations in a class-based storage policy.

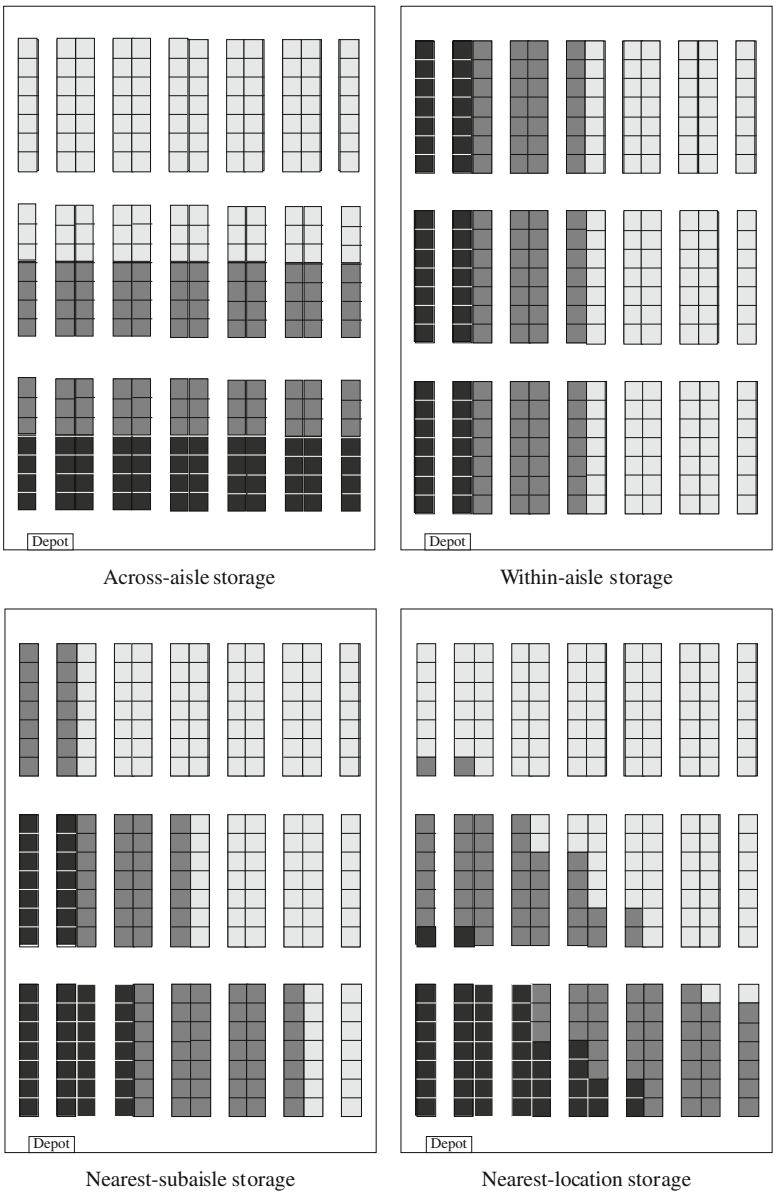


Figure 1.8 – Four ABC-storage assignment methods. Black locations indicate A-items, dark gray locations indicate B-items and light gray locations indicate C-items [Roodbergen, 2012]

Across-aisle storage is a policy where the A-items are assigned to the front-most locations of

each pick aisle. C-items are stored at the back of each pick aisle and B-locations are stored in-between. With the *within-aisle* storage, all products from the same class are in the same aisle (from the front cross-aisle to the rear one). The pick aisles nearest to the I/O (depot) contain the A-items. Jarvis and McDowell [1991] give more details on this policy and obtain good experiment results with it. The *nearest-subaisle* storage assumed that all items in a subaisle belong to the same class, meaning that we have multiple blocks (separating the aisle in 2 or more subaisles). The subaisles with their center closest to the I/O contain the A-items. This is a variation of the within-aisle storage rule. In the *nearest-location* storage, the A-items are assigned to closest locations from the I/O point. One aisle (or subaisle) may contain more than one class. This policy storage has been proved to minimize the travel time of a unit load picking [Petersen, 1999].

For single-block layouts, Jarvis and McDowell [1991] suggest that each aisle should contain only one class, resulting in the *within-aisle* storage policy. Petersen [1999, 2002], Petersen and Schmenner [1999], Petersen and Aase [2004] and Petersen et al. [2004] compare multiple configurations and policies for single-block layouts. The simulation experiments from Petersen and Aase [2004] demonstrates that methods dividing products into classes (such the ABC method) require significantly fewer movements than for random storage. However, if these methods reduce travel, they also increase the administrative workload associated with warehouse management and information management [Petersen et al., 2004]. In addition, multiple pickers who frequently work in the same picking area may generate congestion [Gu et al., 2007]. A system with a large number of orders and multiple pickers will generate a congestion that may slow down the picking operation [Gue et al., 2006]. Pan et al. [2012] present a simulation and approximation methods to compare several storage assignment strategies in presence of congestion with a given order routing strategy.

Finally, a class-based system with one class corresponds to a random storage system and a system with n classes is a full-turnover system, in which n is the number of SKUs.

It is often possible to identify some links between products, such that some of them are regularly ordered together. Products that are frequently ordered at the same time are called correlated or affine [Garfinkel, 2005, Kofler et al., 2014]. It is then straightforward to understand that placing these products near each others will decrease the average travel distance/time. Some call this *family grouping*. When a product from this family of correlated products is out of stock, in sales, etc., other products will be impacted. This introduces some dynamic aspects of storage assignment management, called the dynamic space allocation problem (DSAP) in which managers try to dynamically move products according to a set of insights. For example, consider a set of table and chairs in which the table is backordered. We know that the demand for this set will be lower and we have no interest to keep the chairs of this set in a good position. We can exchange its location with a product that we know will replace this one. It is an example of DSAP with products having affinity.

1.4.3 Order batching

When an order contains a large number of SKUs, it can be picked individually, i.e., a single pick list corresponds to this order. However, when orders are small, it is possible to reduce the travel times by picking a group (batch) of orders at the same time. *Order batching* corresponds to grouping two or more orders in a single picking list [Ruben and Jacobs, 1999, De Koster et al., 2007, Roodbergen and Vis, 2009]. The crucial question is therefore to determine how many orders and which orders to combine into a batched picking list in order to minimize total work and time [Park, 2012] while respecting operational constraints. This problem is a variant of the bin packing problem, making the order batching an NP-hard problem [Gademann and Velde, 2005].

Since this problem is hard to solve to optimality, many researchers have worked on heuristic procedures. For instance, Chen and Wu [2005] use clustering heuristics to batch orders with the aim of maximizing similarity of orders within a batch. There exists two main ways to regroup orders with similar characteristics. The first one assigns each order to a batch based on the distance between each other. This is what we call *proximity batching*. Zhang et al. [2016] present an updated literature list of heuristic methods for the batching problem. The most popular proximity batching heuristics is the *seed* algorithm [De Koster et al., 2007]. The seed algorithm constructs batches in two phases: seed selection and order congruency. The seed selection has a great impact on the given solution. We can select the seed order with different rules: random order, the largest one, longest estimated pick tour, the farthest one, etc. De Koster et al. [1999b] present a large set of seed selection rules. The seed algorithm in an automated environment is considered in Hwang and Lee [1988], Gibson and Sharp [1992], Pan and Liu [1995] and Ruben and Jacobs [1999]. For a manual order picking system, this heuristic is considered in Rosenwein [1994], De Koster et al. [1999a], Ruben and Jacobs [1999], Chen and Wu [2005], Ho and Tseng [2006] and Ho et al. [2008].

Another heuristic rule is the *time window batching*. It is used when the picking operation is made at the same time as orders enter the system (often with due dates). In this case, orders with the same arriving time (or the same due date) are grouped in the same pick list. In most cases, the bin-packing *first-fit* rule is used, but different sorting rules are also presented in De Koster et al. [2007]. The time-window batching is considered in Tang and Chew [1997], Chew and T. [1999], Le-Duc and De Koster [2007], Gong and De Koster [2008] and Lu et al. [2016] with a fixed number of orders per batch and stochastic order arrivals. Henn [2012] and Bukchin et al. [2012] adapt some determinist batching approaches for a dynamic environment with a fixed time period length. These researches do not consider due dates for orders. When due dates are considered, the main objective is to minimize the total tardiness of all orders. Elsayed et al. [1993] and Elsayed and Lee [1996] consider the batching problem aiming at minimizing the tardiness.

Several recent studies have been made on metaheuristic frameworks which aim to improve the batching solution (from the seed or time-windows approaches). Pan et al. [2015] and Koch and Wäscher [2016] present two genetic algorithm approaches, Matusiak et al. [2014b] work on a simulated annealing, Henn and Wäscher [2012] with a tabu search and Henn and Schmid [2013] with an iterated local search algorithm. Žulj et al. [2018] present a hybrid of adaptive large neighborhood search and tabu search. For a more complete list of metaheuristics used for the batching problems, see Zhang et al. [2016].

1.4.4 Routing/sequencing of picking

Until now, we have presented problems impacting the order picking performance, but we have not yet explained how pickers move across the warehouse and aisles. They follow a given route (or sequence) given by a pick list. This pick list determines a sequence of locations that reduces the travel time (or any other criteria). In picker-to-parts systems, this problem of visiting a given set of locations as quickly as possible is a variant of the travelling salesman problem (TSP), which is another NP-hard problem [Lawler et al., 1985, Laporte, 1992]. Knowing locations of a set of SKUs and the distances between each other, it is possible to create a routing graph as in the TSP. The distance computation remains different because of the physical layout of the DC where pickers move within aisles. In a special case of TSP with narrow aisles (no lateral movement within an aisle) and one block, Ratliff and Rosenthal [1983] show that we can use a dynamic-programming method to solve the problem in linear running time with the number of aisles and the number of pick locations. This method has been extended by De Koster and Van der Poort [1998] to determine the shortest order picking routes in a warehouse of one block with a decentralized depot (example: deposit products at the end of each aisle). Roodbergen and De Koster [2001b] extended again this method and developed an algorithm for a warehouse with two blocks (one cross-aisle). Clearly, the distance will be longer when the distance between each side of the aisle is non-negligible (wide aisle). Hall [1993] and Goetschalckx and Ratliff [1988] demonstrate that it still easy to obtain optimal solutions for the traversal of a single aisle. Some classical heuristic methods are presented in Hall [1993], Petersen [1997] and Roodbergen [2001] in the case of one-block warehouse. Most popular methods are presented in Figure 1.9. The bold squares represent SKUs to be picked, and the dashed arrow as the routing path of the picker.

The first heuristic, and probably the most used one, is the *S-shape* heuristic also called the traversal heuristic. It is the simplest one to apply in practice. Routing pickers by using the *S-shape* method means that any aisle containing at least one pick is traversed entirely. An aisle without picks is not entered (it is the same thing for all the other heuristics). From the last visited aisle, the order picker returns to the depot. Another simple heuristic is the *return* method in which the picker enters and leaves an aisle from the same end.

In the *mid-point* heuristic we divide the warehouse into two halves. Picks on the front side

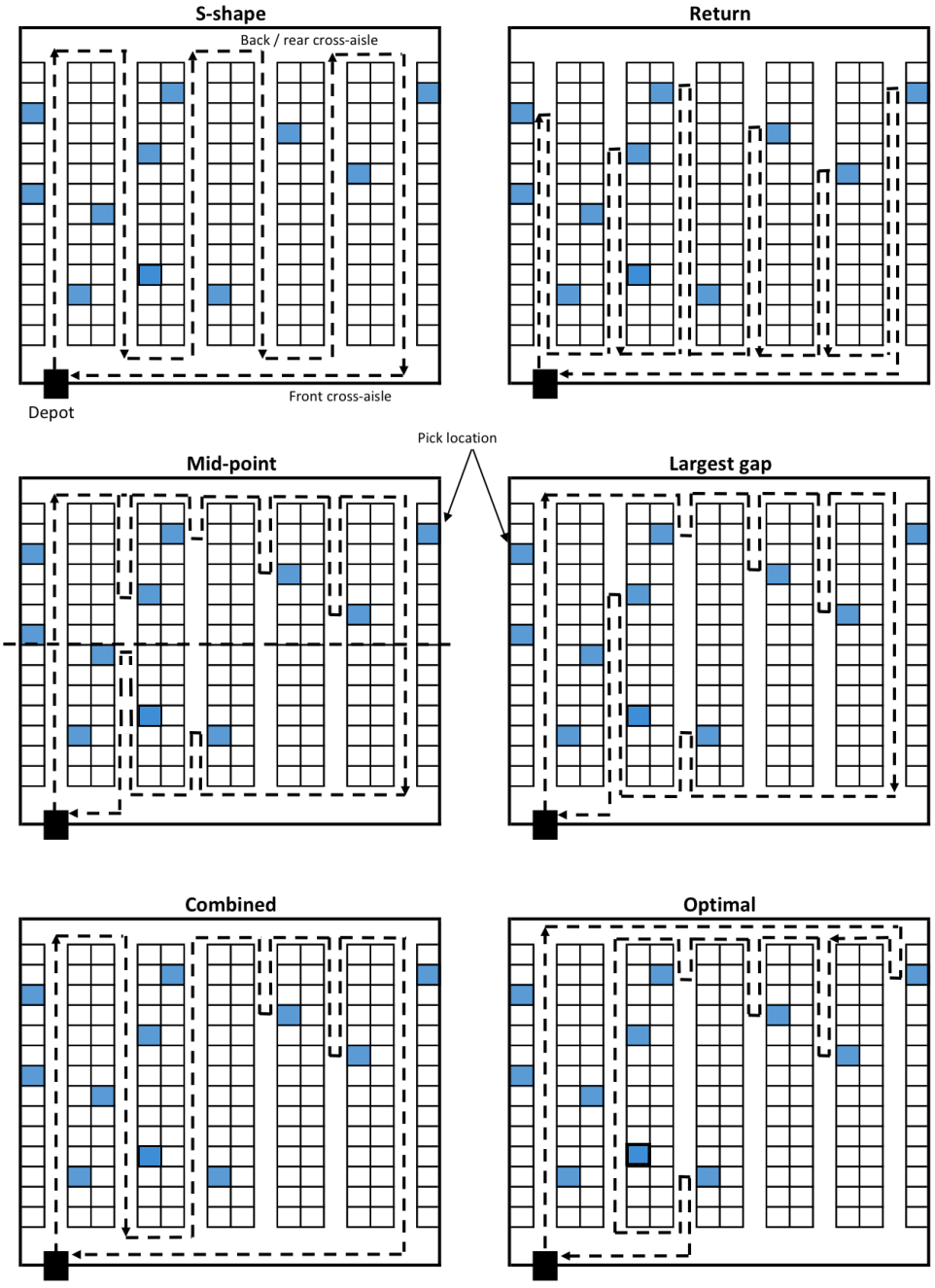


Figure 1.9 – Classical heuristic methods for the sequencing problem [Roodbergen, 2001]

are reached by the front aisle, and picks on the back side reached from the rear aisle. The picker only crosses through the first and last aisles of the warehouse containing products from the pick list. Intermediate aisles are never crossed completely. According to Hall [1993], this method performs better than the *S-shape* when the number of picks per aisle is small.

In the *largest gap* heuristic, the picker follows the same idea as the *mid-point*, but instead of traversing the aisle up to the middle, he enters the aisle up to the item that will leave the *largest gap* for the next item in that aisle [Petersen, 1997]. This heuristic tries to maximize the parts of the aisles that are not traversed (and which have no reason to be). The picker returns and leaves the aisle from the same side (back or front) used to enter it. Again, the first and last aisles are completely crossed. The *largest gap* method outperforms the *mid-point* method [Hall, 1993] in most cases. However, from the implementation point of view, the *mid-point* method is simpler.

The *combined* heuristic approach is based on the composite heuristic of Petersen [1997]. It combines the best features of the return and traversal strategies. It minimizes the travel distance between the farthest picks in two adjacent aisles, and determines for each aisle whether it is shorter to travel the aisle entirely (*S-shape* strategy) or to make a turn in it (the *return* strategy). For the *combined* method, aisles with picks are either entirely traversed or entered and left at the same end. For each visited aisle, the choice is made by using dynamic programming. The reader is directed to Roodbergen and De Koster [2001b] and Roodbergen [2001] for a more comprehensive study of these methods. The combined heuristic is extended to adapt to multi-block warehouses by Roodbergen and De Koster [2001a], named the combined$^+$ in which we force to visit aisles from the right to the left in the block closest to the starting point. The combined$^+$ also allows the picker to start within another than the most left aisle and use the middle aisle afterwards.

Another classical heuristic is the aisle-by-aisle method as presented in Vaughan and Petersen [1999]. Order picking routes resulting from this heuristic visit every pick aisle exactly once and come back to the front aisle (never use the rear aisle). All items in the first aisle are picked, then all items in pick aisle 2, and so on. When there is a middle (cross) aisle, dynamic programming is used to determine the best cross-aisles to take to pass from an aisle to another.

Since there exists several routing methods for the order picking, it is important to know which one performs better and in which conditions. Hall [1993] considers a single-block warehouse where products are randomly placed. He compared the largest gap and the S-shape heuristics and concludes that the largest gap performs better with a pick density fewer than 3.8 (average number of picks per aisle). With a greater density the S-shape outperforms the other method. Petersen [1997] extends this study within the same environment, but compare the six methods presented in Figure 1.9 (except that the combined heuristic is replaced with the composite one, since it was developed later). He concludes that the best heuristic solution (no matter which one is) is on average 5% over the optimal solution and the best overall heuristic procedures are the composite and largest gap methods, which were 9% to 10% over the optimum. De Koster and Van der Poort [1998] and De Koster et al. [1998] use simulations to compare the optimal and S-shape methods for several single-block random storage warehouses. They find that the S-shape provides routes which are, on average, between 7% and 33% longer than the optimum

solutions. Le-Duc and De Koster [2005, 2007] compare the S-shape and return methods in two-block warehouse. They show that the return method gives better results with short pick lists.

Roodbergen and De Koster [2001a] perform a large comparison of six routing methods (optimal, largest gap, S-shape, aisle-by-aisle, combined and combined$^+$), in 80 warehouse instances with different numbers of aisles, cross-aisles and pick list size. These authors show that the combined$^+$ gives the best results in almost all instances. The gaps between the results from the combined$^+$ and the optimal method are large in the case of many aisles and/or large pick-list sizes; they vary between 1% and 25%. They conclude that there is no robust heuristic that is good for all situations. A specific heuristic may be good for one situation but may perform poorly in other situations. This is clearly illustrated in Chapter 3 in which those methods are tested on a heavy constrained environment and perform poorly. We can mention that according to the quality of TSP algorithms, the routing problem is no more really relevant for such classical warehouses with no operational constraints.

Optimizing order picking problems sequentially may yield a suboptimal overall warehouse performance [Van Gils et al., 2017b]. Recently, the benefits of solving the order batching and the routing in a more integrated way were demonstrated in Scholz and Wäscher [2017] and Valle et al. [2017]. Scholz et al. [2017] and Matusiak et al. [2017] present mathematical and heuristic approachs for a joint order batching, assignment and routing problem. For more information about the importance of combining order picking planning problems and its benefit, see Van Gils et al. [2017b].

1.5 Reassignment of products

As presented in Section 1.4.2, the storage assignment policy has a great impact on the whole order picking operation. Nevertheless, these policies need to be regularly reviewed to adapt to changing demand. Most order picking publications consider demand as known in advance. As warehouses accept late orders, the assumption of a constant given demand is questioned in Van Gils et al. [2017a]. This is due to seasonality, products replacement or marketing efforts as presented in Carlo and Giraldo [2012]. As reviewed in Kofler et al. [2014], most studies in this area have been devoted to re-warehousing, involving extensive rearrangement of all locations. This is why this technique is rarely applied in practice. For many managers, finding an optimal products assignment plan is not as important as finding a good one that requires the reassignment of just few products. This second technique is called *healing*.

The study of workload to switch between two assignment has been introduced by Christofides and Colloff [1973]. They propose a two-stage algorithm to minimize the total travel time required to rearrange products within the picking area. More recently, the problem has been revisited by Pazour and Carlo [2015], labelled as the *reshuffling* concept. The reasignment

problem has similarities with the pickup and delivery problem, in which the operator has to collect or drop a product at a location. This is presented by Schrotenboer et al. [2017] in which the authors incorporate the restocking of returned products in the order-picking routes.

In Chapter 4 we show a new idea and formulation for a faster reassignment of products. It shows that significant improvements can be achieved in comparison with observable real-life re-warehousing methods.

1.6 Conclusion

This chapter presented an overview of warehousing operations and functions. More specifically, it described the order picking problem which is a critical operation of the distribution center. There was a strong potential for improvement at this stage of order preparation. Several elements influencing the performance of order picking have been described, such as the layout design of the warehouse, the storage assignment policy, the order batching strategy and the routing method. For each, a series of references are provided showing the evolution of research on the subject. We also observe that, regarding routing methods, current developments in TSP algorithms make the classical problem of routing and sequencing (Chapter 1.4.4) much less relevant. However, when realistic characteristics are considered, the problem becomes more relevant to practise and more complex to solve. These real-life attributes can be related to weight, capacity or intrinsic characteristics of SKUs imposing new and challenging constraints on the problem.

Chapter 2

Mathematical model, heuristics and exact method for order picking in narrow aisles

Résumé

Ce chapitre est motivé par une collaboration avec un partenaire industriel oeuvrant dans la vente et distribution de meubles et électroménagers. Nous avons modélisé leur problème de préparation de commandes en allées étroites comme une variante du problème de tournées de véhicules avec diverses adaptations de la matrice de distance. Les problèmes de sécurité présents en allées étroites imposent un degré supplémentaire de difficulté lors de la détermination des itinéraires. Nous montrons que négliger les aspects dimensionnels (2D) et résoudre le problème sur un entrepôt unidimensionnel produit une différence significative. Nous avons résolu un grand nombre d'instances reproduisant des configurations réalistes en utilisant une combinaison d'heuristiques et d'un algorithme exact, minimisant la distance totale de prélèvement. Grâce aux expérimentations, nous identifions les méthodes les mieux adaptées à chaque configuration d'allée. Nous comparons nos solutions avec celles obtenues par les procédures de l'entreprise, montrant que des améliorations peuvent être obtenues.

Chapter information A paper based on this chapter is published in *Journal of the Operational Research Society*: Chabot T., Coelho L.C., Renaud J. and Côté J.-F. Mathematical model, heuristics and exact method for order picking in narrow aisles. Journal of the Operational Research Society, Dec 2017, Pages 1-12. This research was also the subject of a presentation at the joint CORS/INFORMS Conference in Montréal, Canada (2015).

Abstract

This chapter is motivated by a collaboration with an industrial partner who delivers furniture and electronic equipment. We have modeled their narrow aisles order picking problem as a vehicle routing problem through a series of distance transformations between all pairs of locations. Security issues arising when working on narrow aisles impose an extra layer of difficulty when determining the routes. We show that these security measures and the operator equipment allow us to decompose the problem per aisle. In other words, if one has to pick orders from three aisles in the warehouse, it is possible to decompose the problem and create three different instances of the picking problem. Our approach yields an exact representation of all possible picking sequences. We also show that neglecting 2D aspects and solving the problem over a 1D warehouse yields significant difference in the solutions, which are then suboptimal for the real 2D case. We have solved a large set of instances reproducing realistic configurations using a combination of heuristics and an exact algorithm, minimizing the total distance traveled for picking all items. Through extensive computational experiments, we identify which of our methods are better suited for each aisle configuration. We also compare our solutions with those obtained by the company order picking procedures, showing that improvements can be achieved by using our approach.

2.1 Introduction

Distribution centers (DCs) play a central role in modern supply chains which are characterized by an increasing number of different products, or stock-keeping units (SKUs), and by more frequent but smaller orders. Considering that order delivery within 24 hours is becoming a new industry standard, the performance of order picking operations in DCs is of critical importance. Because order picking is labor intensive for most warehouses, the design and control of warehouse order picking systems are highly strategic decisions De Koster et al. [2007], Renaud and Ruiz [2008], Roodbergen and Vis [2006], Roodbergen et al. [2008], Rouwenhorst et al. [2000]. This chapter is based on a collaboration with a Québec-based (eastern Canada) company which operates in the furniture and electronics industry. This industry is characterized by a very large number of SKUs since many products can be customized (type and color of woods and leathers, large variety of furniture fabrics, etc.) and by a large number of orders, each having few items. To efficiently deal with such an ordering pattern and product variety, the DC of our partner is organized in warehousing zones. Large electrical appliances are stacked directly on the floor; expensive electronic devices are stored in controlled access sections; finally, general furniture products, which represent about 70% of the total number of SKUs, are stored in narrow aisles with single-depth shelves on both sides. Such mixed-width aisle configuration is useful to optimize both space utilization and order picking productivity [Mowrey and Parikh, 2014]. Figure 2.1 shows a sketch of our partner's furniture storage zone, where we can see the reception docks (rear), the aisles, the shipping area (front), and the direction of the flow.

Pallets are unloaded from the incoming vehicles (reception area) with standard lift trucks and products are placed in front of their corresponding aisle. Then, an operator with a turret forklift puts the products in place. Narrow aisle lift trucks operate only in the front part of the DC and thus are imposed to enter and exit the aisles from the front. For the expedition, when exiting an aisle, order pickers store products in the shipping area before being loaded in the shipping vehicles. Since each narrow aisle truck must enter and leave each aisle by the same side, and since a picker cannot go back to another aisle with products, the order picking process is separable by aisle (a truck can just enter an aisle when it is empty).

Warehouse control presents numerous challenges, namely how to operate receiving, storing, order picking, replenishment and shipping Bodnar and Lysgaard [2013], De Koster et al. [2007], Gu et al. [2007]. Storage and order picking are the two most intensively studied operations. Storage addresses how the warehouse is divided and how space is allocated to products; order picking is related to how products are picked to fulfill customer orders. Some of the most important issues in order picking are batching procedures and routing methods De Koster et al. [2007], Hong et al. [2012], Petersen [1997, 1999], Wruck et al. [2013].

Routing methods have already been used in several variations of the order picking problem.

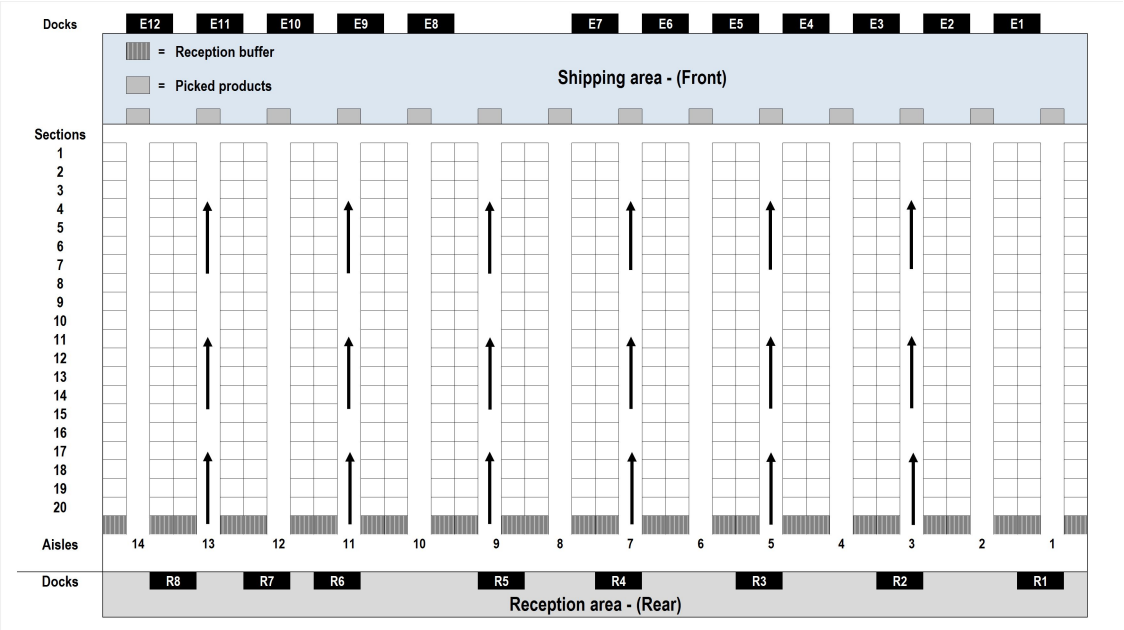


Figure 2.1 – General overview of a warehouse with narrow aisles

Goetschalckx and Ratliff [1988] showed that the optimal aisle traversal can be modeled and solved as a shortest path problem. Single order picking in a general warehouse reduces to a standard traveling salesman problem (TSP). Ratliff and Rosenthal [1983] showed that order picking in a rectangular warehouse is a special solvable case of the TSP. Roodbergen and De Koster [2001b] developed an algorithm to find the shortest picking tour in a parallel aisle warehouse with a middle aisle. Kim et al. [2003] produced routes for a gantry around a large warehouse dedicated to cosmetic products. Despite these special cases, most realistic order picking problems are much more complex as they need to handle complicating constraints [Chabot et al., 2017b]. Among these real-life constraints is the use of narrow aisles, which limits lift trucks circulation.

Narrow aisles are useful to maximize floor space utilization, and they facilitate the picking process as the picker has direct access to the shelves on both sides of the aisle [Gue et al., 2006]. However, they require the use of special narrow aisle man-up turret lift truck, which must be driven according to some specific rules enforced for security reasons. In narrow aisles warehouses, traffic often becomes an important issue [Hong et al., 2012, Parikh and Meller, 2010], but in our case traffic problems are eliminated by the DC practices according to which replenishments are performed during the daytime while order picking and truck operations are performed at night. Also, each picker is in charge of a specific number of aisles. These are

some standard procedures used in this industry which make the problem at hand significantly different from other industries.

In our problem, individual orders arrive during the day and are accumulated until the evening, when the order picking process starts. This means that we deal with one big picking list per aisle, consisting of the aggregation of many smaller orders. Picking routes are then defined to collect all the products. The subset of products grouped into the same route defines a batch, for which the picking sequence must be determined. Usually, the order batching problem (OBP) [De Koster et al., 2007, Henn et al., 2012] uses a heuristic to estimate the distances traversed for the picking process; we, on the other hand, determine the optimal routing. When all products are picked they are sorted to rebuild the initial orders. In this process, one particular order may be split between different routes, as well as a route may consolidate many orders.

In most situations, the number of orders and products to pick is too high to allow for the simultaneous optimization of order batching and picking sequence. Thus, it becomes necessary to determine the batching while estimating the distance of the sequencing problem to be solved, which is known to be an NP-hard problem [Gademann and Velde, 2005]. There are few studies addressing exact algorithms for the OBP. Gademann et al. [2001] presents a branch-and-bound algorithm for the OBP that minimizes the total travelling time for picking all orders. The performance of an order batching policy is closely related to that of the picking. Öncan [2015] presents mixed integer linear programming formulations for the order batching under three different policies: traversal, return and midpoint. He also develops an efficient local search procedure based on tabu search for these problems. Gademann and Velde [2005] and Muter and Öncan [2015] show that the OBP can be modeled using a Set Partitioning Problem, for which it is possible to develop branch-and-price algorithms. The benefits from solving the order batching and the routing in a more integrated way are demonstrated in Scholz and Wäscher [2017] and Valle et al. [2017].

In this chapter we describe the *Capacitated Narrow Aisle Order Picking Problem* (CNA-OPP) in narrow aisles. A picker can move forward/backward, and up/down. Products are readily available within reach of the picker on the right and on the left. Since no movement is associated with this action, we say the problem is defined over 2D. To the best of our knowledge it is the first time that this problem is considered with the multi-dimensional aspect. Through a series of computations of the distance matrix between all pairs of products of the 2D warehouse, and taking new and complex practical constraints into account, we show how to model it as a well-known vehicle routing problem (VRP) Semet et al. [2014]. By considering the 2D physical properties of the problem we develop an effective arc reduction procedure which allows us to solve the model faster and with smaller gaps. We derive a heuristic to reproduce the company decision-making procedure in order to accurately compute the cost of their current solutions. In addition, we develop an adaptive large neighborhood search (ALNS) heuristic as well as

an exact branch-and-cut algorithm to solve realistic instances to optimality within very short computing times. Our algorithms are specifically tailored to take advantage and exploit the new structure of the model.

The remainder of the chapter is organized as follows. In Section 2.2 we formally describe the CNA-OPP. Section 2.3 shows how to model this problem as a VRP taking into consideration the structure of the warehouse and security rules. Section 2.4 presents the algorithms we have developed to solve the problem. The results of extensive computational experiments of our algorithms and that of the company are detailed in Section 2.5, where we also provide a comparison of different layouts. Our conclusion is presented in Section 2.6.

2.2 Problem description

The CNA-OPP is defined as follows. The warehouse is composed of narrow-aisles arranged with two single-depth racks having s sections long and t levels high. A set $\mathcal{P} = \{1, \ldots, m\}$ of m SKUs has to be picked up from a given aisle. The location of SKU i in the aisle can be defined in a Cartesian coordinate system (x_i, y_i, z_i) where $x_i \in \{1, \ldots, s\}$, $y_i \in \{1, \ldots, t\}$, and $z_i \in \{1, 2\}$, where z_i represents the left or right rack of the aisle. We assume that x_1 is the first section of the aisle and x_s is the last one. The input/output (I/O) point of each aisle is modeled as being the position $(0, 0, 0)$. Note that the $(0, 0, 0)$ is the only location in the section 0. Each SKU i has a weight w_i in kilograms and a volume v_i in cubic feet. When performing a picking sequence, the maximum weight and volume that can be put on a pallet are W and V, respectively. All locations in the rack are identical, the horizontal distance between two consecutive sections is α_h, and the vertical distance between two consecutive levels is α_v. The parallel distance between left and right racks is equal to the aisle width and to that of the picking vehicle. Thus, the lateral distance is negligible and will be considered as zero. Because all m SKUs have to be picked up and given that the handling time is identical for all products, it can be ignored without loss of generality.

In order to ensure picker safety, two security rules must be respected. First, any horizontal lift truck movements, i.e., change of sections, must be performed at the ground level. This is a classical safety rule generally used in most warehouses. The second constraint is that after having performed a pick, other truck movements should be in the forward direction to avoid products falling on the picker and because he will lose visibility on the rear side as presented in Figure 2.2. Since section numbers are ordered such that the largest number corresponds to the farthest section and the picks must be done by getting closer to the (I/O), a movement from i to j may take place if $x_i > x_j$. Note that it is allowed to pick at different levels and on both sides of the aisle because this does not imply horizontal lift truck movements.

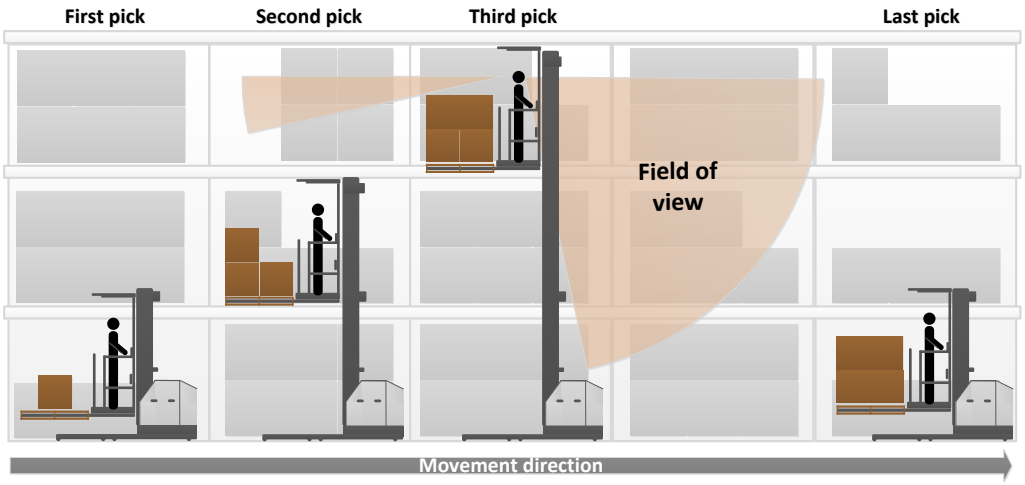


Figure 2.2 – Illustration of equipment and physical constraints

2.3 Distance matrix and mathematical formulation

In this section we provide the formal mathematical description of the problem. The definition of the distance matrix is provided in Section 2.3.1. The mathematical formulation is introduced in Section 2.3.2, and the procedure used to eliminate variables of the problem is presented in Section 2.3.3.

2.3.1 Computing the distance matrix

We define a distance matrix $\mathcal{D} = d_{ij},\ i, j = 0, \ldots, m$, over the I/O point and all product locations. Recall that $(0, 0, 0)$ is the only feasible location at the section 0, i.e., the I/O point. In order to respect safety constraints, $\mathcal{D}$ is defined as:

$$d_{ij} = \begin{cases} \alpha_h x_j + \alpha_v y_j & \text{if } x_i = 0 \quad (2.1a) \\ \alpha_v |y_i - y_j| & \text{if } x_i = x_j > 0 \quad (2.1b) \\ \alpha_h (x_i - x_j) + \alpha_v (y_i + y_j) & \text{if } 0 < x_i > x_j \quad (2.1c) \\ \infty & \text{otherwise.} \quad (2.1d) \end{cases}$$

Case (2.1a) computes the distances when starting from the I/O point to section x_j and level y_j. This is the only group of distances towards the rear of the warehouse. Case (2.1b) computes the distance between two locations in the same section as the sum of vertical distances, i.e., the difference between the levels. Note that this condition imposes a distance equal to zero from a location to itself. Case (2.1c) is the more general case in which the picker changes sections, going from section x_i to x_j, with $x_i > x_j$. This distance is composed of two segments: the distance between the two sections ($\alpha_h(x_i - x_j)$) and the sum of the two vertical movements

$(\alpha_v(y_i+y_j))$. The latter movement is the distance down to the bottom of the rack and then the distance up to the appropriate location in the next section. Case (2.1d) prohibits movements from section x_i to x_j if $x_j > x_i$.

For example, consider a simple order with only two SKUs. The first pick i is at section $x_i = 5$, level $y_i = 4$ and on the left aisle, i.e., z_i=1, so at $(5, 4, 1)$, and the second pick j is at section $x_j = 2$, level $y_j = 3$ and right side, thus at $(2, 3, 2)$. Starting from the I/O point, the picker must first perform the pick of section 5 and level 4, go to the floor level, move to section 2, and then go up to level 3. The first movement from the I/O point to product i incurs a distance of $(5\alpha_h + 4\alpha_v)$. Traveling from product i to product j incurs a distance $(5-2)\alpha_h + (4+3)\alpha_v$. Returning from j to I/O point requires a distance of $(2\alpha_h + 3\alpha_v)$, for a total distance of $(10\alpha_h + 14\alpha_v)$. Note that if the picker decides to pick the second item first, he will not be able to move backward to pick the first one in the same trip. In this case, two trips from I/O point are required with a total distance of $(14\alpha_h + 14\alpha_v)$.

2.3.2 Mathematical model

The CNA–OPP consists of determining the order picker routes, starting and ending at the I/O point, such that all products are picked and the traveled distance is minimized while respecting capacities and safety constraints. It is modelled over a graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ where $\mathcal{V}$ $= \{0, 1, \ldots, m\}$ is the set of vertices in the aisle with a picking demand, composed of the m location nodes $\mathcal{V}'= \{1, 2, \ldots, m\}$ assigned to specific requests with weight w_j and a volume v_j. We define the arc set $\mathcal{A} = \{(i, j) : i, j \in \mathcal{V}, i \neq j, i = 0, \ldots, m, j = 1, \ldots m\}$ and distance d_{ij} associated with arc (i, j).

In order to define a transportation plan that respects the practical constraints of this problem and minimizes the total work of the picker, we define the following decision variables. Let u_{ij} be a binary variable equal to 1 if arc (i, j) is used, and k be an integer variable indicating the number of routes (picking tours) needed. The CNA-OPP can then be modelled as follows:

$$\text{minimize} \sum_{(i,j)\in\mathcal{A}} d_{ij}u_{ij} \tag{2.2}$$

subject to

$$\sum_{j\in\mathcal{V}} u_{0j} = k \tag{2.3}$$

$$\sum_{i\in\mathcal{V}} u_{i0} = k \tag{2.4}$$

$$\sum_{j\in\mathcal{V}\setminus\{i\}} u_{ij} = 1 \qquad \forall i \in \mathcal{V} \tag{2.5}$$

$$\sum_{i \in \mathcal{V} \setminus \{j\}} u_{ji} = 1 \qquad \forall j \in \mathcal{V} \tag{2.6}$$

$$\sum_{i \in S} \sum_{j \in S} u_{ij} \leq |S| - r(S) \qquad S \subseteq \mathcal{V}', |S| \geq 2 \tag{2.7}$$

$$k \geq r(\mathcal{V}') \text{ and integer} \tag{2.8}$$

$$u_{ij} \in \{0, 1\}. \tag{2.9}$$

The objective function (2.2) minimizes the total distance traveled. Constraints (2.3) and (2.4) ensure that the same number of routes will leave and return to the I/O point. Constraints (2.5) and (2.6) are degree constraints and ensure that all products will be picked up exactly once, while (2.7) simultaneously forbids subtours and ensures that the volume capacity is respected. Here, $r(S) = max\left\{\left\lceil \frac{\sum_{j \in S} v_j}{V} \right\rceil, \left\lceil \frac{\sum_{j \in S} w_j}{W} \right\rceil\right\}$. Movement restrictions do not need to be imposed through constraints as they will be avoided due to the values from the distance matrix. Constraints (2.8) indicates the lower bound of the variable k as the minimal number of vehicles needed to serve the total demand. Constraints (2.9) define the nature of variables.

2.3.3 Arc reductions

We now present three procedures to remove several arcs from the graph and significantly decrease its size. The first one is due to the restriction of the picker to move from lower to higher sections. For security reasons, all arcs from a lower section to a higher section can be removed as these paths are not feasible.

The second procedure considers products located in the same section and at the same level, but on different sides. Since the picker can pick the two items in any order, this creates symmetric solutions. We can thus remove all arcs (i, j) with $z_i < z_j$.

The last procedure considers products from the same section but at different levels. Recall that the picker always starts and ends a section at ground level. This structure enables the removal of arcs between products in the same section. Let S be a set of products located on different levels of the same section. Among all paths visiting all the products of S only two are of minimal distance. The first path collects products from the lowest level to the highest level. The second one does the opposite, collecting products from the highest level to the lowest one. For example, if three products are located on levels 3, 7 and 11 then paths (3, 7, 11) and (11, 7, 3) are clearly the least costly. Other paths such as (7, 3, 11) will require additional vertical movements. To remove all more expensive paths, and at the same time subtours within the same section, all arcs from a product located in a lower level to a higher level in the same section can be removed.

To summarize our three rules, arc (i, j) with $i \in \mathcal{V} \setminus \{0\}$ is removed if any of the following applies:

1. $x_i < x_j$;
2. $x_i = x_j$, $y_i = y_j$ and $z_i < z_j$;
3. $x_i = x_j$ and $y_i < y_j$.

Proposition 1. *The use of all these procedures eliminates all cycles in the graph that do not contain the I/O point and thus prevents subtours.*

Proof By contradiction, suppose all procedures were applied and at least one cycle is present. It means there exists a path from a product i located in a lower section, or at a lower level, or having a lower index than another product j. However, because all procedures were applied, no arcs exist from i to k where k is in a farther section, or at a higher level or having a higher index. Consequently, the graph does not contain any cycles. □

These reductions substantially simplify the complexities of this new type of VRP, hereinafter referred to as *acyclic VRP* (Ac-VRP). However, the Ac-VRP remains an $\mathcal{NP}$-hard problem. To prove this, we show that any instance of the *bin-packing problem* (BPP) can be polynomially transformed into an instance of the Ac-VRP.

Definition 1. *Bin Packing Problem.* Given a set of identical bins of capacity $\bar{W}$, and a set of $\bar{n}$ items each having weight $\bar{w}_j \leq \bar{W}$ $(j = 1, 2, ..., \bar{n})$, the problem is to assign all items to bins using a minimal number of bins while ensuring that the sum of the weights of the items assigned to a bin does not exceed its capacity $\bar{W}$.

Proposition 2. *The Ac-VRP is $\mathcal{NP}$-hard.*

Proof Consider an instance of the BPP with bins of capacity $\bar{W}$ and $\bar{n}$ items of weight $\bar{w}_j \leq \bar{W}$ $(j = 1, 2, ..., \bar{n})$. A node is created in the Ac-VRP for each item of the BPP instance. The node associated with the item $j = 1$ is located in the farthest section, the second item in the second farthest section and so on. There is an arc of cost zero that is created for each pair of nodes (i, j) in which $i > j$.

For each node, two arcs are created: one of cost 1 from the depot to the node and one of cost 0 from the node to the depot. This graph yields an Ac-VRP because it does not contain any cycles. Since the BPP is $\mathcal{NP}$-hard, the Ac-VRP is also $\mathcal{NP}$-hard. □

2.3.4 Reason to consider 3D aspects

The previous section presented some techniques to remove a large set of arcs. Since a number of 2D-related arcs have been eliminated, one may be tempted to solve the problem in 1D.

Solving this problem using a 1D plan corresponds to putting all products at the ground level and ignoring the vertical distance between the levels. In other words, one may argue that there is no need to consider the problem in 2D and simply solve it in 1D assuming its optimal solution is also an optimal solution for the original 2D problem. Figure 2.3 shows a counter-example where the optimal 1D solution does not lead to an optimal 2D one. Here, consider a small warehouse with only three sections and three levels. The starting point of each picking route is on the left. For simplicity, let all vertical and horizontal distances be equal to one. The distance between each side of the aisle remains zero. The distance between the I/O point and the first section is one unit. The vertical distance with the first level remains zero. In the figure, we split the aisle down along its sections.

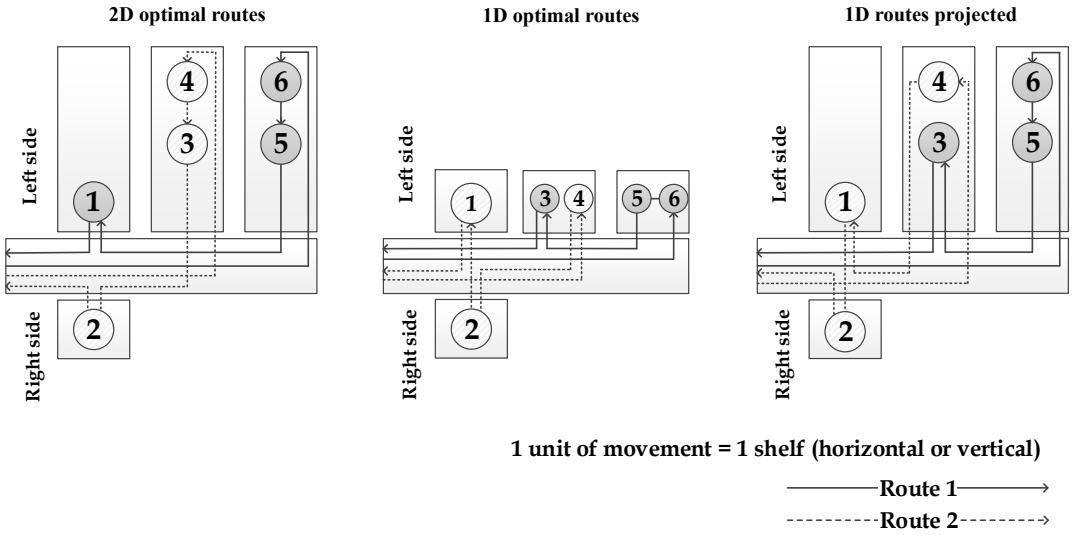


Figure 2.3 – Example of 2D optimal solution projected on a 3D plan

The first part of the figure represents the optimal 2D solution. In its first route (solid line), product 6 is picked up first, traversing 3 units horizontally and 2 units vertically, following to products 5 and then 1, for a total of 10 units of distance. The second route (dashed line), picks products 4, 3 and 2, in that order, for a total distance of 8 units, totalling an optimal solution for the 2D case with a cost of 18 units of distance. The middle part of the figure shows an optimal 1D solution containing the same products placed at the ground level, thus ignoring the vertical distance that represents the 2D aspect of the problem. This solution costs 10 units of distance in 1D. However, projecting its routes on the 2D space, as is shown in the third part of the figure, yields a suboptimal solution: route 6-5-3 has a cost of 12 and route 4-1-2 has a cost of 8, for a total of 20 units of distance, which is larger than the optimal solution depicted in the first part of the figure. It is easy to observe that the more empty vertical shelves the warehouse contains, the worse the impact of neglecting its 2D aspect is. This clearly motivates the work on the distance matrix that considers the multidimensional

property of this problem.

2.4 Solution methods

Since the CNA-OPP can be modelled as a CVRP over an asymmetric graph, we have implemented three different algorithms for its resolution. In Section 2.4.1 we describe the order picking method used by our industrial partner. Section 2.4.2 presents an adaptive large neighborhood search heuristic, and in Section 2.4.3 we present a branch-and-cut algorithm.

2.4.1 Actual picking method used by the company

In order to compare our results with those obtained by our industrial partner, we have implemented a greedy heuristic that reproduces their current picking method. For each picking route, they start with the product in the farthest section at the highest level. Then, they add the next feasible product from the next highest level in the same section. If no more products can be feasibly picked in the same section, they go back to the farthest next section where a product can be picked. In this way they always choose the product in the farthest section at the highest level, which respects the security movement constraints. The pseudocode for this set of rules is presented in Algorithm 1.

Algorithm 1 Picking method used by the company

1: Sort the set of requests L such that the following two conditions hold for all $i, j \in L$ $x_{L[i]} \geq x_{L[j]}$ and if $x_{L[i]} = x_{L[j]}$ then $y_{L[i]} \geq y_{L[j]}$
2: **while** L is not empty **do**
3: Create a new picking route R with a total volume $D = 0$ and total weight $P = 0$
4: Initialize route with $L[1]$
5: $D = D + v_{L[1]}$ and $P = P + w_{L[1]}$ and $L = L \setminus L[1]$
6: **for** request i in the sorted L **do**
7: **if** $D + v_{L[i]} \leq V$ and $P + w_{L[i]} \leq W$ **then**
8: Add request $L[i]$ to the route R
9: $D = D + v_{L[i]}$ and $P = P + w_{L[1]}$ and $L = L \setminus L[i]$
10: **end if**
11: **end for**
12: **end while**

First, we use a list L of products which is ordered by sections, and for the same section it is sorted by levels. For a same level, ties between left and right aisle are broken arbitrarily (line 1). Then, we create a new route R, and start by picking the farthest product $L[1]$, i.e., the one at the farthest section at the higher level (lines 3 and 4). The route is completed by picking the next closest item or until the capacity constraints are reached, always respecting the safety movement restrictions (lines 6 to 10). When an item is added to a route, it is removed from the list L and all following items in L are moved up once. When the route is full, another route is created and initialized again with the farthest item from the remaining items from L.

This approach is optimal if the total volume and weight of all the products to be picked up is less or equal to the capacity V and W of the picking vehicle.

2.4.2 Adaptive large neighborhood search

We present an implementation of an ALNS heuristic for our problem, widely based on Ropke and Pisinger [2006]. The ALNS is composed of a set of simple destruction and reconstruction heuristics in order to find better solutions at each iteration. One of the general strengths of ALNS comes from the hybridization between the simulated annealing and a variable neighborhood search that provides a great diversity in the local search. From the order picker perspective, this method provides fast picking routes minimizing the travel time and increase the picked order rates.

The ALNS selects one of many destroy and repair operators at each iteration. We have implemented three destroy and two repair operators. Our destroy operators include the Shaw removal [Shaw, 1997], the worst removal, and a random removal. Our repair operators include a greedy parallel insertion and a k-regret heuristic [Potvin and Rousseau, 1993]. Each operator is selected with a probability that depends on its past performance and a simulated annealing acceptance criterion is used. The idea of the Shaw removal operator is to select a set of somewhat similar items (relatedness). In our case, the relatedness of two nodes is based on the distance between them. Products in the same section are more related to each other. The worst removal operator compares the cost of the solution with and without an item. It removes items with a higher impact on the value. The greedy insertion operator is a construction heuristic inserting unplaced items at their minimum cost position over all routes. The regret heuristic operator finds items that maximize a regret value. The regret value is the cost of inserting the item in its best route and its second-best route.

The regret heuristic is used to build the initial solution. The ALNS is ran for 50 000 iterations of destroy-repair operators. After every 100 iterations, the weight of each operator is updated according to its past performance. Initially, all the operators have the same weight. A sketch of our ALNS is provided in Algorithm 2 and for further details about operators and parameterization, we refer to Ropke and Pisinger [2006].

2.4.3 Branch-and-cut algorithm

We have implemented a branch-and-cut algorithm in which valid linear inequalities are used as cutting planes to strengthen a linear programming relaxation at each node of a branch-and-bound tree. Constraints (2.7) are initially relaxed and only added to the branch-and-bound tree if they are found to be violated. The structure of our formulation is based on the vehicles routing problems. That is why we opt for this resolution method which is highly recognized to be effective for this type of problem structure.

Algorithm 2 Adaptive large neighborhood search

```
Create an initial solution S;
S* ← S;
Initiate probability ρ^d for destroy operators and ρ^i for repair operators;
while stop criterion is not met do
    Select a number of picks 1 ≤ q ≤ n;
    Select a removal and insertion operators using ρ^d and ρ^i;
    Apply operators on S to obtain S';
    if f(S') < f(S) then
        S ← S';
        If f(S') < f(S*) then S* ← S';
    end if
    if f(S') ≥ f(S) then
        S ← S' according to simulated annealing criterion;
    end if
    Update ρ^d and ρ^i;
end while
return S*.
```

The whole model can be solved by feeding it directly into an integer linear programming solver to be solved by branch-and-bound if the number of rounded capacity inequalities (7) is not excessive. However, for instances of realistic size, e.g., 30 or 40 nodes, as is the case of our partner, the number of rounded capacity constraints (7) is too large to allow full enumeration and these must be dynamically generated throughout the search process. The exact algorithm we present is a branch-and-cut scheme in which the rounded capacity inequality constraints are generated and added into the program whenever they are found to be violated. It works as follows. At a generic node of the search tree, a linear program containing the model with a subset of the subtour elimination constraints and relaxed integrality constraints is solved, a search for violated inequalities is performed, and some of these are added to the current program which is then reoptimized. This process is reiterated until a feasible or dominated solution is reached, or until there are no more cuts to be added. At this point, branching on a fractional variable occurs. We provide a sketch of the branch-and-cut scheme in Algorithm 3.

2.5 Computational results

The algorithms described in Section 2.4 were implemented in C++. The branch-and-cut uses the CVRPSEP library Lysgaard [2003] and IBM CPLEX Concert Technology 12.7 as the branch-and-bound solver. All computations were executed on machines equipped with two Intel Westmere EP X5650 six-core processors running at 2.667 GHz, and with up to 48 GB of RAM installed per node running the Scientific Linux 6.3. We have first solved each instance using the ALNS algorithm. This solution was then used to start the branch-and-cut, to which a time limit of 7200 seconds was imposed.

Algorithm 3 Branch-and-cut algorithm

```
Subproblem solution: Solve the LP relaxation of the current node
Termination check
if there are no more nodes to evaluate then
    Stop
else
    Get the proposed node from CPLEX branch-and-bound
end if
while solution of the current LP relaxation contains subtours do
    Identify connected components with CVRPSEP Lysgaard [2003]
    Add violated subtour elimination constraints
end while
if the solution of the current LP relaxation is integer then
    Go to the termination check
else
    Branching: branch on one of the fractional variables
    Go to the termination check
end if
```

In order to evaluate the performance of our algorithms and that of the company, we have generated instances inspired by real warehouse configurations. First we have generated five types of aisle configurations as presented in Table 1. The second and the third columns show respectively the number of sections and levels present in an aisle. The fourth column shows the total number of locations according to this configuration. Finally, we indicate the ratio of the layout between sections and levels. These configurations will allow us to assess the impact of different aisle configurations. For each type of configuration, 10 sets of instances having each 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100 requests were generated. For each of the five aisle configurations and for each of the 10 number of requests, we generated five instances. Thus, 50 instances were generated for each aisle configuration for a total of 250 instances. The typical operation of our industrial partner consists of orders containing fewer than 30 requests. We have generated instances with more requests to assess the performance of our methods.

The products are chosen randomly from the database obtained from our partner along with their volume and weight and placed randomly on the aisle. Our instances can contain more products to pick than the total number of locations, meaning that products can be picked more than once.

We provide in Table 2.2 the average results for the method used by the company and for our algorithms on all 250 instances. The results are organized by the number of requests, thus each row presents the average of 25 instances for all five aisle configurations. The first column shows the number of picks, followed by the solution value yielded by the company heuristic, then the solution of our ALNS algorithm, its running time in seconds, the best known solution yielded by CPLEX for both formulations, its lower bound, percentage gap and running time

Table 2.1 – Types of aisle configuration

Type	Sections	Levels	Locations by side	Total locations	Ratio
1	12	4	48	96	3
2	24	4	96	192	6
3	36	4	144	288	9
4	48	4	192	384	12
5	60	4	240	480	15

in seconds. To validate our graph reduction method, we provide the results of the cyclic and the acyclic models.

It is clear that the ALNS algorithm is very effective and run in a few seconds even for the largest instances. Moreover, CPLEX using ALNS solution as initial start value is able to marginally improve the solution obtained by this algorithm. We observe an improvement for instances starting from 30 requests for both formulations. Finally, as is known from other VRP problems, a branch-and-cut algorithm is capable of proving optimality or yielding tight optimality gaps for instances containing fewer than 100 nodes, as is corroborated by our experiments.

As it can be observed in the *Company* column of Table 2.2, which represents their current picking method, this heuristic rarely yields the optimal solution or yields solutions as good as those of the ALNS algorithm. The company heuristic yielded only 33 optimal solutions (out of 250), 31 of them being for the smallest instances with 20 or fewer picks. The global average of the company method is 4.58% worse than the solutions obtained with the ALNS algorithm. In a large warehouse being operated daily over a long period, this represents a large reduction in the workload. Using the ALNS solution as a starting point for the CPLEX algorithm seems to be a good approach to help the solver in reaching optimality. The *Time (s)* column shows how ALNS is a fast heuristic given the large number of iterations we have allowed. We see that on average, the acyclic version is faster and obtains better lower bounds and has reduced the overall average gap from 0.51% to 0.28%.

In Table 2.3 we show the gap between the heuristic methods and the lower bound (LB). These lower bounds are given by CPLEX after two hours of running time and correspond to the best LB obtained by either models for each instance. With the branch-and-cut algorithm, we observe that it was able to solve to optimality instances with up to 100 requests. We see that all instances with 50 or fewer requests were solved to optimality with the acyclic graph, and that the number of optimal solutions decreases when the number of requests increases. In total, our cyclic model solved 169 instances and the acyclic model solved 192 instances to optimality from our set of 250 instances.

For the company heuristic, the gap with the best lower bound ranges between 0.28% and 6.47% with an average of 4.07%. This shows that ALNS is able to yield better solutions, even

Table 2.2 – Average results ordered by the number of picks

# **picks**		ALNS		Cyclic formulation				Acyclic formulation			
	Company	Sol.	Time (s)	UB	LB	Gap (%)	Time (s)	UB	LB	Gap (%)	Time (s)
10	282.88	282.08	0.68	282.08	282.08	0.00	0.00	282.08	282.08	0.00	0.04
20	517.84	509.84	0.48	509.84	509.84	0.00	0.08	509.84	509.84	0.00	0.04
30	692.24	674.80	0.92	673.92	673.92	0.00	29.12	673.92	673.92	0.00	0.76
40	889.76	861.44	1.24	860.40	860.40	0.00	308.60	860.40	860.40	0.00	6.64
50	1010.08	969.76	2.04	969.76	969.12	0.06	1215.20	969.76	969.76	0.00	138.92
60	1147.52	1089.68	2.84	1088.56	1084.93	0.44	2527.20	1089.04	1087.28	0.19	1465.48
70	1395.76	1322.80	3.88	1322.56	1315.82	0.62	4736.40	1321.68	1317.87	0.28	2858.00
80	1529.12	1451.68	5.24	1451.28	1442.44	0.76	4553.40	1450.64	1444.89	0.48	2780.36
90	1738.64	1640.88	6.72	1640.80	1618.41	1.59	6315.92	1640.64	1628.01	0.87	5691.72
100	1886.96	1779.44	8.44	1779.36	1755.04	1.65	5826.40	1779.04	1762.24	1.00	5253.32
Average	**1109.08**	**1058.24**	**3.25**	**1057.86**	**1051.20**	**0.51**	**2551.23**	**1057.70**	**1053.63**	**0.28**	**1819.53**

Table 2.3 – Average percentage gap of the heuristics and exact method

# picks	Company	Gap (%) with respect to best LB	ALNS	Gap (%) with respect to best LB	B&C best LB	# optimals cyclic	# optimals acyclic
10	282.88	0.28	282.08	0.00	282.08	25	25
20	517.84	1.54	509.84	0.00	509.84	25	25
30	692.24	2.65	674.80	0.13	673.92	25	25
40	889.76	3.30	861.44	0.12	860.40	25	25
50	1010.08	3.99	969.76	0.00	969.76	23	25
60	1147.52	5.25	1089.68	0.22	1087.28	17	21
70	1395.76	5.48	1322.80	0.26	1319.31	9	17
80	1529.12	5.43	1451.68	0.39	1446.01	11	16
90	1738.64	6.27	1640.88	0.68	1629.71	4	6
100	1886.96	6.47	1779.44	0.82	1764.88	5	7
Average	**1109.08**	**4.07**	**1058.24**	**0.26**	**1054.32**	**169/250**	**192/250**

for small instances. With respect to the best lower bound obtained by CPLEX, the ALNS provides an average gap of 0.26%. We note that these solutions were hardly improved by the exact method.

Since the CNA-OPP is subject to movement restrictions, we must analyze the impact of the layouts configuration and the performance of the algorithms. With these constraints on section movements, we observe that different configurations can have a significant impact on the computation performance. The bigger it is, the more it can contain different products and have a longer distance for picking. In the same way, an instance with many requests in a small aisle, will contain the same products appearing multiple times. Therefore, it is difficult to compare the layout based only on the total distance. However, it is possible to measure the performance of methods for each type of aisle configuration based on the ratio between the number of sections and levels.

In Table 2.4 we present the average gap over five instances yielded by the solver after two hours, for each type of layout and all numbers of picks, according to the aisle configuration presented in Table 2.1. When all instances are solved to optimality, we indicate it by a zero in bold. We can observe that the average gap (over 50 instance for each column) is well correlated to the ratio sections/levels outlined in Table 1. This means that for the same number of requests, optimal solutions are easier to obtain with aisles having more sections.

Both versions are still influenced by the number of sections in the aisle. Based on our algorithms, the easier aisle configuration is that with 60 sections (Layout 5). With the acyclic model on layout 5, we have obtained an average CPLEX gap of 0.19% and with the cyclic model and the layout 5, this average gap is 0.17%.

For the acyclic version of the model, we notice that layouts 4 and 5 are the best in terms of performance. Layout 4 combined with an acyclic graph solves instances up to 80 requests to optimality. All other configurations are not able to solve to optimality all instances of the sample with more than 80 requests. Based on our set of instances, it is easier for CPLEX to solve problems in an aisle with more sections. The worst configuration for this group of instances is the first one, with only 12 sections. According to the movement restrictions, it seems difficult to create high quality tours in a short aisle. Obviously, a longer aisle will yield solutions with longer routes, but for which our methods can obtain solutions of better quality.

Table 2.5 presents computational details of both models in terms of CPU times, optimality gap, number of cuts, number of branch-and-bound nodes and number of variables. As seen in Table 2.2, the acyclic formulation is able to close the gap faster than the cyclic formulation on average. We see in Table 2.5 that the average number of visited node for the cyclic formulation is 8669.04 versus 2298.66 for the acyclic, around 3.77 times fewer nodes. It also shows that the we only remove a very small number of variables to pass from the cyclic with an average of 2081.22 variables, to the acyclic formulation with 2007.99. It represents only 4% of arcs

Table 2.4 – Average percentage gap for each type of layout

# picks	Cyclic model					Acyclic model				
	Layout 1	Layout 2	Layout 3	Layout 4	Layout 5	Layout 1	Layout 2	Layout 3	Layout 4	Layout 5
10	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
20	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
30	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
40	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
50	**0.00**	0.20	**0.00**	**0.00**	0.11	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
60	1.18	0.59	0.35	**0.00**	0.06	0.63	**0.00**	0.20	**0.00**	0.13
70	1.35	0.50	0.32	0.69	0.27	0.29	**0.00**	0.76	0.35	**0.00**
80	1.12	1.29	1.01	0.23	0.16	0.57	1.02	0.59	**0.00**	0.24
90	3.12	0.86	2.61	0.65	0.72	1.70	0.76	0.71	0.48	0.71
100	3.11	1.23	1.48	2.07	0.38	1.44	0.97	0.66	1.11	0.83
Average	**0.99**	**0.47**	**0.58**	**0.36**	**0.17**	**0.46**	**0.27**	**0.29**	**0.19**	**0.19**

reduction. Even if the reduction yields better results, we see that the number of added cuts are larger for the acyclic formulation.

Table 2.5 – Models analysis

# of picks	Cyclic formulation					Acyclic formulation				
	CPU (sec)	Gap (%)	Cuts	Nodes	Variables	CPU (sec)	Gap (%)	Cuts	Nodes	Variables
10	0.00	0.00	2.64	0.00	67.72	0.04	0.00	1.24	0.00	65.96
20	0.08	0.00	45.40	18.04	237.72	0.04	0.00	33.80	3.04	230.92
30	29.12	0.00	752.24	7696.96	512.32	0.76	0.00	411.80	140.88	495.88
40	308.60	0.00	1768.32	6433.04	888.68	6.64	0.00	693.68	205.32	860.72
50	1215.20	0.06	3998.88	16537.12	1372.00	138.92	0.00	2164.28	1267.28	1325.76
60	2527.20	0.44	7722.12	12820.60	1959.48	1465.48	0.19	7642.92	4349.16	1890.36
70	4736.40	0.62	9684.52	19469.44	2650.28	2858.00	0.28	10923.36	7464.88	2555.24
80	4553.40	0.76	7494.96	11644.92	3444.80	2780.36	0.48	6755.88	2107.96	3320.36
90	6315.92	1.59	9276.92	6930.12	4340.08	5691.72	0.87	15545.44	4125.32	4185.20
100	5826.40	1.65	8752.96	5140.12	5339.12	5253.32	1.00	12896.80	3322.76	5149.52
Average	**2551.23**	**0.51**	**4949.90**	**8669.04**	**2081.22**	**1819.53**	**0.28**	**5706.92**	**2298.66**	**2007.99**

2.6 Conclusion

In this chapter we have proposed a formulation for the CNA-OPP that yields a well-known CVRP. We have then been able to use CVRP algorithms to solve this difficult problem arising in warehousing operations. Based on the problem structure, we have shown that the number of arcs can be reduced leading to an acyclic model which is easier to solve, despite still being NP-Hard. We have also shown that one must consider the 2D aspect of the problem in the solution algorithm, as neglecting this feature and solving the problem with a projection of the products on the floor level will significantly decrease solution quality once the cost is evaluated over the 2D distance. With a collaboration from an industrial partner, we have generated a large dataset of benchmark instances for the order picking problem based on real data. We have proposed five configurations of an aisle and tested one heuristic and one exact algorithm against the picking method used by the company. We have shown that improvements of up to 4.07% can be obtained by using our methods.

Chapter 3

Order Picking Problems under Weight, Fragility, and Category Constraints

Résumé

Dans ce chapitre, nous modélisons et résolvons un problème complexe de prélèvement de commandes motivé par nos observations dans le secteur de la distribution alimentaire. Ce problème combine des décisions complexes de prélèvement et de routage dans le but de minimiser la distance parcourue. Nous fournissons une description de l'entrepôt qui nous permet de calculer algébriquement les distances entre toutes les paires de produits. Nous proposons ensuite deux modèles mathématiques distincts pour formuler le problème. Nous développons et adaptons cinq méthodes heuristiques, dont une implémentation d'une métaheuristique. Nous implémentons ensuite un algorithme exact pour résoudre les formulations. La performance des méthodes proposées est évaluée sur nouvelle banque d'instances. Des expérimentations approfondies confirment l'efficacité de la métaheuristique. Nous confirmons l'hypothèse selon laquelle les heuristiques classiques ont du mal à trouver une solution efficace dans un environnement hautement contraint.

Chapter information A paper based on this chapter is published in *International Journal of Production Research*: Chabot T., Lahyani R., Coelho L. C. and Renaud J. Order picking problems under weight, fragility and category constraints. International Journal of Production Research, June 2016, Pages 6361-6379. This research was the subject of two presentations: at the 58th CORS conference in Banff, Canada (2016) and at the 17th ROADED Conference in Compiègne, France (2016).

Abstract

Many practical constraints arising in real-life have often been neglected in the order picking routing literature as presented in Chapter 1. In this chapter, we model and solve a rich order picking problem under weight, fragility, and category constraints, motivated by our observation of a real-life application arising in the grocery retail industry. This difficult warehousing problem combines complex picking and routing decisions under the objective of minimizing the distance traveled. We first provide a full description of the warehouse design which enables us to algebraically compute the distances between all pairs of products. We then propose two distinct mathematical models to formulate the routing problem. We develop five routing heuristic methods, including extensions of the classical largest gap, mid-point, S-shape, and combined heuristics. The fifth one is an implementation of the powerful adaptive large neighborhood search algorithm specifically designed for the problem at hand. We then implement a branch-and-cut algorithm and cutting planes to solve two formulations. The performance of the proposed solution methods is assessed on a newly generated test bed containing up to 100 pickups and seven aisles. Extensive computational experiments confirm the efficiency of the ALNS metaheuristic and derive some important insights for the order picking in this kind of warehouse. We hence confirm one of the thesis hypothesis saying that classical heuristics struggle to find an effective solution within highly constrained environment, since they were not tailored for this. Then, we need a new approach reaching better order picking distances.

3.1 Introduction

Order picking represents an important part of warehouse and inventory management activities [Gu et al., 2010a]. It consists of retrieving goods from specific storage locations to fulfill customers orders. It is considered as the most labor-intensive and costly warehousing activity [De Koster et al., 2007, Tompkins et al., 2010], and it may account for up to 55% of all warehouse operating expenses [Chiang et al., 2011]. Therefore, warehouse productivity is highly affected by order picking activities [De Koster et al., 2007]. Research in this area has rapidly expanded over the past decades. These studies can be categorized into books [Hompel and Schmidt, 2006, Ackerman, 2013, Manzini, 2012], survey papers [Wäscher, 2004, De Koster et al., 2007, Henn, 2012], and theoretical papers providing mathematical formulations and exact or approximate solution methods [Bortolini et al., 2015, Lu et al., 2016].

Order picking strategies and algorithms have often been studied for classical warehouses [Petersen, 1997, Petersen and Aase, 2004]. The role of picker personality in predicting picking performance with different picking technologies has been studied by De Vries et al. [2016]. Recently, some attention has been devoted to researches oriented towards more realistic picking contexts [Chackelson et al., 2013, Matusiak et al., 2014a, Chabot et al., 2017a]. These cases are either motivated by the complex characteristics of real-life warehousing activities, legal regulations, or the introduction of on-line shopping requiring faster and more customized services.

This chapter is motivated by the situation prevailing in the grocery retail industry. In this industry, each company owns one or several distribution centers (DCs) to store its products. Almost all DCs are designed with reserve storage and fast pick areas. In the fast pick area, each stock keeping unit (SKU) is stored in a specific and dedicated location. However, in the reserve storage area, SKUs are placed at random available locations, but close to the picking and sorting area. Order pickers only have access to the pick area, which is replenished continually by employees from the reserve area. Each order is picked in a sequential way and transported to the sorting area by hand-lift trucks. Each picker deals with a list of products, i.e., a pick-list, to be collected from the storage aisles.

In general, when orders are small compared to the picking vehicle capacity, order batching occurs. Order batching consists of combining orders to reduce the distance traveled by the picker [De Koster et al., 2007, Petersen, 1997, 1999]. Batched orders must later be sorted. In our case, each individual order may consist of several hundreds of different SKUs, often resulting in thousands of units picked on many pallets.

The sequences followed by the pickers to retrieve all the SKUs of a given order are called routes. A fixed routing plan may perform well for some pick-lists, but poorly for others as decisions are made sequentially. Thus, the primary and most common objective in manual order-picking systems is to find the set of picking routes that minimizes the total traveled

distance [De Koster and Van der Poort, 1998, De Koster et al., 1998, Petersen et al., 2004].

In this chapter we develop order picking routes in a warehouse under practical restrictions observed in the grocery retail industry. Each product characteristic considered adds an extra layer of difficulty to the problem. These product-specific properties are described next.

We introduce the order picking problem under weight, fragility, and category constraints (OPP-WFCC). Regarding the weight, as soon as the total weight of all the products transported on the hand-lift truck exceeds a threshold value, *heavy items* can no longer be collected, and the picker is only allowed to pick *light* SKUs. Thus, another tour is required to pick (some of) the remaining heavy items. This requirement, referred to as *weight* constraint, has two motivations. It helps avoid work accidents and back problems caused by lifting heavy charges to relatively high positions, and it ensures the vertical and horizontal stability of the pallet. Besides the weight, other constraints arise in practice regarding the fragility of the items. A product can support a certain weight without being crushed. Therefore, fragile products must not be placed underneath heavy products, i.e., heavy products must be on the bottom of the pallet and light products on the top. We refer to this restriction as *fragility* constraint and the maximum weight a product can support is referred to hereinafter as *self-capacity*. Finally, we consider two types of commodities: food and non-food products. Non-food products encompass household items. Food products should not be carried under non-food on the pallet in order to avoid contamination. Thus, one has to pick the non-food products separately or before any food products. This constraint is referred to as *category* constraint.

Order picking problems dealing with the physical properties of the products have not been widely studied in the literature, but similar constraints appear in other contexts. Matusiak et al. [2014a] refer to these constraints as precedence constraints since they impose that some products must be picked before some others due to weight restrictions, fragility, shape, size, stackability, and preferred unloading sequence. They propose a heuristic method to solve the joint order batching and picker routing problem without any specific assumption regarding the layout and without any pre-determined sequencing constraints. Junqueira et al. [2012] introduce the problem of loading rectangular boxes into containers while considering cargo stability and load bearing constraints. Their mathematical model ensures the vertical and horizontal cargo stability and limits the maximum number of boxes that can be loaded above each other. Dekker et al. [2004] solved a real-world application arising in a warehouse storing tools and garden equipment. The authors considered different assumptions arising in this particular real-life application such as the design of the warehouse with non-coinciding start and end points, dead-end aisles, and two floors. For the sake of efficiency improvement, the authors examined the storage and the pickup policies of these products while ensuring that heavy products are picked first to prevent damaging fragile products. They proposed storage heuristics and routing heuristics to consider the application restrictions.

Routing problems with precedence constraints, in which one request must be served and/or loaded before another, have been widely studied in the VRP literature and may arise in several real-life contexts. Some practical applications include the dial-a-ride problem [Cordeau and Laporte, 2007], airline scheduling [Medard and Sawhney, 2007], and bus routing [Park and Kim, 2010]. If one introduces capacity constraints to the VRP, depending on the precedence constraints, the resulting problem is a pickup and delivery [Zachariadis et al., 2016]. For more details on the VRP with precedence constraints, see Lahyani et al. [2014]. The multi-constrained OPP studied in this paper is similar but quite different from the VRP with precedence constraints. Indeed, considering the self-capacity and the fragility constraints adds an extra layer of difficulty to the problem since it has a significant impact on the storage and routing strategies.

The contributions of this chapter are threefold. First, we introduce a rich variation of the order picking problem under weight, fragility, and category constraints inspired from the grocery retail industry. Part of the complexity of the problem is due to the new practical constraints arising from the separation of the products, their fragility, the stability, and weight limit of the pallet. The second contribution is the development of two distinct formulations to model the problem, which include a precise computation of the distance matrix between all pairs of items within the warehouse and the development of original valid inequalities. Our third contribution is to develop heuristic algorithms and different exact methods to support warehouse picking operations in choosing the most suitable routing sequences to satisfy orders. Specifically, we propose branch-and-cut as well as an adaptive large neighborhood search (ALNS), and an extension of four classical order picking algorithms to solve the OPP-WFCC. We then solve large sets of instances reproducing realistic configurations using our algorithms, which aim at minimizing the total distance traveled for picking all items. We show that the classical order picking heuristics in the literature do not perform well for constrained order picking problems.

The remainder of the chapter is organized as follows. In Section 3.2 we formally describe the problem and define its particularities. Section 3.3 presents two mathematical models along with a set of new valid inequalities. The details of the five heuristic algorithms and of the branch-and-cut procedures are provided in Section 3.4. The results of extensive computational experiments are presented in Section 3.5, and our conclusions follow in Section 3.6.

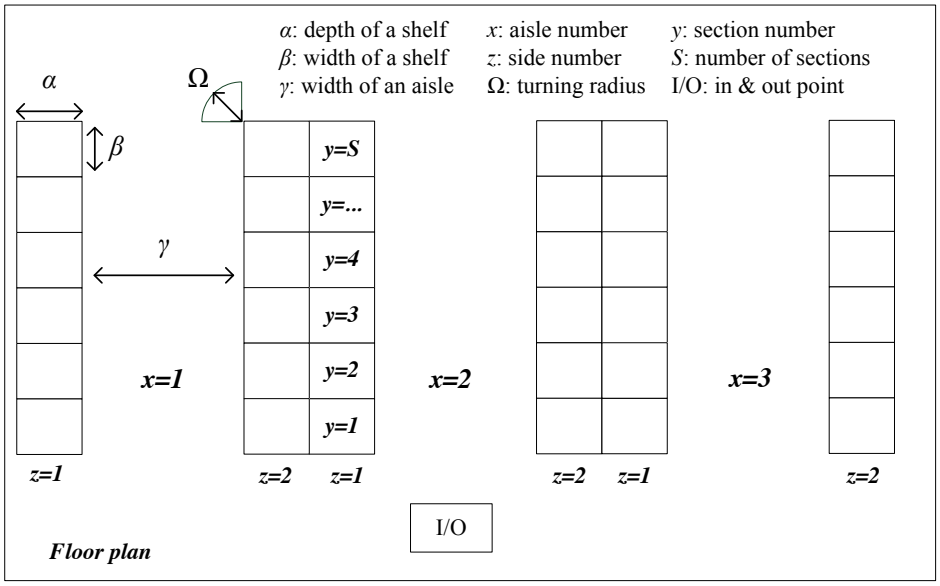
3.2 Problem description and distance modeling

In order to properly model the problem, which aims to minimize the total distance traveled by the pickers, one needs to know precisely the distances between all pairs of positions within the warehouse under study. We have modeled a warehouse constituted of several parallel aisles on a plane. There are complete racks in the middle of the warehouse and two half-racks on either side. These aisles are perpendicular to two corridors, one in the front and one in the rear.

This configuration is commonly used and is generic enough for the purpose of this chapter. Besides, the location of a product is fixed and each product takes just one location. In other words, each product exists in only one location within the warehouse. Finally, we suppose that the distance between the two sides of an aisle is large, such that the picker cannot pick items from both sides at the same time. We also suppose the one-dimensional stacking. The warehouse has one input/output (I/O) point which is considered as the departure and the arrival point for all the pickers. In addition, the warehouse is symmetric with respect to the I/O, with the same number of aisles on either side.

The distance between different locations is computed by solving a shortest path problem. Observe that in order to change aisles, the picker can move through the rear or the front corridor, and these two paths usually yield different distances. Let x_i represent the aisle number of SKU i, and y_i its section number, where $y_i \in \{1, \ldots, S\}$. The total number of sections S represents the number of in-depth locations of the aisle. A warehouse with one aisle and 20 sections per aisle contains 40 locations, by considering both sides. Note that aisles with vertical picking are modeled as well, but it is not a common layout in the grocery industry. The higher rack levels contain bulk stock, which is used to replenish the ground level, which contains the pick stock of the items. The side of product i within an aisle is z_i, with $z_i \in \{1, 2\}$. Using this notation, one can fully represent the location of product i as its coordinates (x_i, y_i, z_i) [Chabot et al., 2017a]. The width of an aisle is γ units of distance, the depth of a location is α units of distance, and its width is β. Since there are S sections in an aisle, the total length of the aisle is βS. The minimal distance between two aisles is given by 2α. In order to consider the complex characteristics of the real-world application under study, we consider a turning radius, denoted Ω. These features are visualized in Figure 3.1.

Figure 3.1 – Overview of the warehouse layout in the grocery retail industry



We define the distance matrix $\mathcal{D} = d_{ij}$ where $i < j$ and $i \in \{0, \ldots, m-1\}$, $j \in \{1, \ldots, m\}$, over the I/O point and all m product locations. Several scenarios must be considered when defining $\mathcal{D}$ as follows.

(i) Assume that i and j refer to two distinct products, then two cases may arise. The first one appears when both products are in the same aisle. The distance d_{ij} between i and j is then:

$$\underset{i<j}{d_{ij}} = |y_i - y_j|\,\beta + |z_i - z_j|\,\gamma \quad \text{if } x_i = x_j. \tag{3.1}$$

The first term computes the distance between the two sections, and the second term accounts for the aisle width γ if the products are on different sides of the aisle.

The second case arises when the two products are in different aisles. With the aim of easing the notation, we separate the equations into two segments: the length-wise distance, denoted υ, and the width-wise described next. The length proportion is expressed by $\upsilon = \min(\beta(2S - y_i - y_j), \beta(y_i + y_j)) + 2\Omega$. The total distance can then be computed as:

$$\underset{i<j}{d_{ij}} = \begin{cases} |x_i - x_j|\,(2\alpha + \gamma) + \upsilon & \text{if } z_i = z_j \quad (3.2a) \\ (x_j - x_i)(2\alpha + \gamma) + \gamma + \upsilon & \text{if } z_i = 1, z_j = 2 \quad (3.2b) \\ (x_j - x_i)(2\alpha) + (x_j - (x_i + 1)) + \upsilon & \text{if } z_i = 2, z_j = 1. \quad (3.2c) \end{cases}$$

The first case appears when products are on same side, but in different aisles. In the second case, the items are in two different sides, such that the picker has to cross the aisle width at the departure and at the arrival aisles. In the third case, products i and j are placed such that the start and arrival aisles are not crossed.

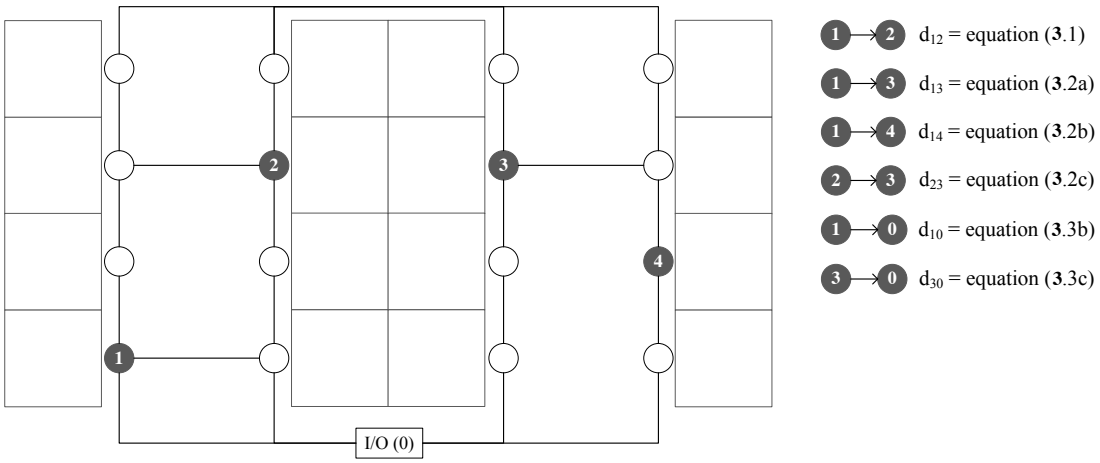
(ii) Now, assume that i refers to a product and j refers to the I/O point. The distances between a product and the I/O point are symmetric and must be computed by taking into account three cases. The first and easiest one is when section x_i of item i coincides with section x_0 of the I/O point. The second appears if $x_i < x_0$, and the third if $x_i > x_0$. These distances are then computed as:

$$d_{i0} = \begin{cases} y_i\beta + \dfrac{1}{2}\gamma & \text{if } x_i = x_0 \quad (3.3a) \\ (x_0 - x_i)2\alpha + (x_0 - x_i - z_i + 1)\gamma + \dfrac{1}{2}\gamma + \Omega + y_i\beta & \text{if } x_i < x_0 \quad (3.3b) \\ (x_i - x_0)2\alpha + (x_i - x_0 + z_i - 2)\gamma + \dfrac{1}{2}\gamma + \Omega + y_i\beta & \text{if } x_i > x_0. \quad (3.3c) \end{cases}$$

In the first case, in which $x_i = x_0$, the distance is the total of the number of sections plus half of the width of an aisle. The second case appears when the product is located on the

left of the I/O point, and the third case if the product is on the right. These two cases are symmetrical and are composed of the width of the cells to be traversed, the width of the aisles to be crossed, the turning radius distance, and the length of the sections needed to traverse to reach the product. The different scenarios defining the distance matrix $\mathcal{D}$ are illustrated in Figure 3.2.

Figure 3.2 – Illustration of the distance matrix and the corresponding equations

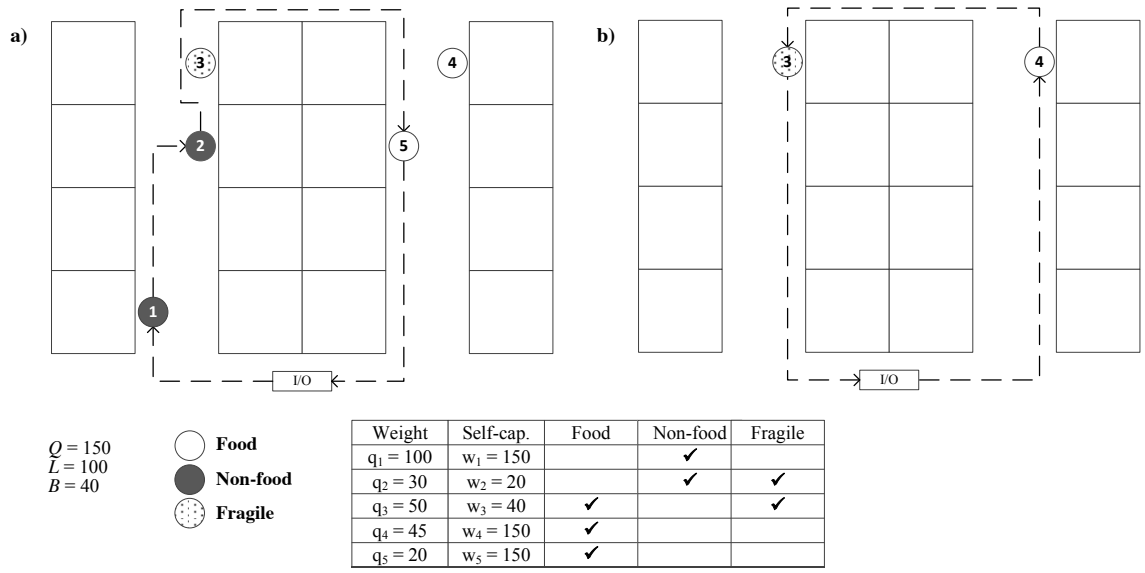


The OPP-WFCC is formally defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ where $\mathcal{V}$ is the vertex set and $\mathcal{A}$ is the arc set. The set $\mathcal{V} = \{0, \ldots, m\}$ contains the location of the I/O point and the m products locations constituting the set $\mathcal{V}' = \{1, \ldots, m\}$. The set $\mathcal{A} = \{(i, j) : i \in \mathcal{V}, j \in \mathcal{V}', i \neq j\}$ is the arc set. Each product $i \in \mathcal{V}'$ has a weight q_i. In the OPP-WFCC products have three important characteristics. First, a product is said to be light if its weight is under B units, otherwise it is considered as a heavy item. Second, a product can also be fragile or non-fragile. A weight limit w_i, i.e., a self-capacity, is associated with each fragile item i. If a given item i is considered as non-fragile, then w_i is assumed to be Q which corresponds to the total weight that can be loaded on a pallet. Finally, products are also categorized as food or non-food items. Non-food items cannot be loaded on top of food items. With this requirement, we observe that arcs (i, j) with i being a food item and j being a non-food item can be removed from $\mathcal{A}$. Finally, when the total weight of all picked items on the pallet reaches a limit L, no more heavy products can be picked in the same tour. The objective of OPP-WFCC is to find the set of tours minimizing the total distance while respecting the weight, fragility, and category constraints.

To better describe the particularities of this problem, in Figure 3.3 we sketch a solution for the OPP-WFCC in which precedence constraints may interfere. The output of the example solution includes two routes where each route begins and ends at the I/O node. The picking points are visited in a specific order respecting the weight, fragility, capacity, and category constraints. Items 1 and 2 are non-food products while items 3, 4, and 5 are food products.

Besides, products 2 and 3 are considered fragile. The first route, presented in Figure 3.3a, starts by picking product 1 then product 2. Since these are non-food products, the picker can load food-products on the top of the stack while respecting the category constraints. However, the total weight of the pallet reaches the threshold $L = 100$ so the picker can no longer load any heavy products. Since the self-capacity of product 2 equals to 20, product 3 ($q_3 = 50$) or product 4 ($q_4 = 45$) cannot be loaded on top of it. Finally route 1 picks product 5 before going back to the I/O node. The products left in the warehouse are 3 and 4. Since product 4 is heavier than the self-capacity of product 3, route 2 first picks product 4, then product 3, and then goes back to the I/O (Figure 3.3b). The set of constraints requires the picker to do two tours, both going around the warehouse and the two aisles. Without constraints, the picker could only make one tour to pick everything. In such a case, the distance to pick all products is almost the double.

Figure 3.3 – Example of a solution for the OPP-WFCC



Weight	Self-cap.	Food	Non-food	Fragile
$q_1 = 100$	$w_1 = 150$		✓	
$q_2 = 30$	$w_2 = 20$		✓	✓
$q_3 = 50$	$w_3 = 40$	✓		✓
$q_4 = 45$	$w_4 = 150$	✓		
$q_5 = 20$	$w_5 = 150$	✓		

3.3 Mathematical formulations

We now provide two different formulations for the OPP-WFCC. In Section 3.3.1 we present a capacity-indexed formulation which makes explicit the remaining capacity of the pallet traversing each arc. In Section 3.3.2 we present a two-index vehicle flow formulation.

Recall that d_{ij} denotes the distance between two nodes i and j defining arc (i, j) computed as described in Section 3.2. Let $\mathcal{V}'_h$ be the set of nodes associated with heavy products.

3.3.1 Capacity indexed formulation

In this section, we propose a new formulation to model the OPP-WFCC, referred to as capacity-indexed formulation. This type of formulation has only appeared a few times for basic and rich variants of VRPs [Picard and Queyranne, 1978, Pessoa et al., 2009, Poggi de Aragão and Uchoa, 2014, Lahyani et al., 2015]. This formulation is compact enough to enumerate all variables and constraints for small and medium instances of the problem. We later incorporate new procedures to further reduce the number of variables. Let binary variables x_{ij}^q indicate that arc (i, j) is used with a remaining capacity of q units. Let $\mathcal{C} \subset Q$ be the subset of possible values of q from 1 to Q. The formulation is defined by:

$$\text{(F1) minimize} \sum_{(i,j)\in\mathcal{A}} d_{ij} \sum_{q\in\mathcal{C}} x_{ij}^q \tag{3.4}$$

subject to

$$\sum_{j\in\mathcal{V}} \sum_{q\in\mathcal{C}} x_{ij}^q = 1 \quad i \in \mathcal{V} \tag{3.5}$$

$$\sum_{i\in\mathcal{V}} \sum_{q\in\mathcal{C}} x_{ij}^q = 1 \quad j \in \mathcal{V} \tag{3.6}$$

$$\sum_{i\in\mathcal{V}} x_{ij}^q - \sum_{k\in\mathcal{V}} x_{jk}^{q-q_j} = 0 \quad j \in \mathcal{V}', q > q_i \in \mathcal{C} \tag{3.7}$$

$$\sum_{i\in\mathcal{V}'} x_{i0}^q = 0 \quad q > 1 \in \mathcal{C} \tag{3.8}$$

$$\sum_{j\in\mathcal{V}'} x_{0j}^0 = 0 \tag{3.9}$$

$$x_{ij}^q \in \{0, 1\} \quad i, j \in \mathcal{V}, q \in \mathcal{C}. \tag{3.10}$$

The traveled distance is minimized in (3.4). Equations (3.5) and (3.6) are the in and out degree constraints. They ensure that each node is visited exactly once. Equations (3.7) guarantee that if the picker visits a node j with a remaining capacity q, then he must leave this node after picking load q_j with a remaining capacity of $q-q_j$. These constraints ensure the connectivity of the solution and the capacity requirements. Infeasible and unnecessary variables are removed with equalities (3.8) and (3.9). Equations (3.8) forbid arcs to return to the I/O point with a remaining capacity different from zero. Similarly, equations (3.9) state that all arcs visiting a node i must carry a load with available space for item i. Constraints (3.10) define the domain of the variables.

The main advantage of formulation F1 is that it enables to preprocess the model to decrease its size and to remove infeasible variables by adding appropriate constraints in the preprocessing

of the variables. Indeed, because of the physical characteristics of the products, precedence constraints should be considered when defining picking route to avoid putting non-food products above food products on the pallet. One can identify arcs going from non-food to food products beforehand. Moreover, one can ensure that the safety requirements for fragile products are always respected by removing infeasible variables. Because the index q carries the information about the remaining weight capacity, all arcs with q greater than the self-capacity w_j and the weight q_j of the destination product j, i.e., $q > w_j + q_j$, cannot exist. Thus, all the arcs generated respect the weights that fragile products can support. We can also eliminate variables x_{ij}^q traversing arcs (i, j) with infeasible remaining capacity, i.e., with a capacity $q < Q - q_i$. In what concerns the *weight* constraint, the picker cannot load any heavy product if the total weight of the pallet reaches the threshold L. We model this restriction with inequalities (3.11):

$$\sum_{i,j \in \mathcal{S}} x_{ij}^q + x_{lh}^q \leq |\mathcal{S}| - 1 \quad \mathcal{S} \subseteq \mathcal{V}' : q_s \geq L, l \in \mathcal{S}, h \in \mathcal{V}_h' \backslash \mathcal{S}. \tag{3.11}$$

Recall that a heavy product is a product whose weight exceeds the limit B. Consider a subset of nodes $\mathcal{S}$ associated with products whose total weight is greater than or equal to L. We check if the potential item to be picked in the same route is a heavy product and we impose that the picker can only pick light items. The arc (l, h) links the last node l of the subset $\mathcal{S}$ and the node corresponding to the heavy product h. Constraints (3.11) can be improved by considering all the potential heavy products that cannot be picked on the same route. We lift the second term of the left-hand side over the nodes related to heavy products, i.e., the nodes in the subset $\mathcal{V}_h'$. Constraints (3.11) can thus be replaced by:

$$\sum_{i,j \in \mathcal{S}} x_{ij}^q + \sum_{h \in \mathcal{V}_h' \backslash \mathcal{S}} x_{lh}^q \leq |\mathcal{S}| - 1 \quad \mathcal{S} \subseteq \mathcal{V}' : q_s \geq L, l \in \mathcal{S}. \tag{3.12}$$

Constraints (3.12) can be further generalized by removing the restriction to leave the subset $\mathcal{S}$ from the last node visited in this subset. We then replace constraints (3.12) by:

$$\sum_{i,j \in \mathcal{S}} x_{ij}^q + \sum_{l \in \mathcal{S}} \sum_{h \in \mathcal{V}_h' \backslash \mathcal{S}} x_{lh}^q \leq |\mathcal{S}| - 1 \quad \mathcal{S} \subseteq \mathcal{V}' : q_s \geq L. \tag{3.13}$$

Since the number of constraints (3.13) is exponential (in the order of $O(2^m)$), they cannot be generated a priori. We propose a branch-and-cut algorithm to add them dynamically. Details about this algorithm are provided in Section 3.4.

It is possible to strengthen the formulation by adding some valid inequalities. Taking into account the physical characteristics of the products and their structural capacity, we can

identify general situations in which a solution is not feasible. We already use several of these characteristics in the variables creation, and we could derive the following new cuts. In constraints (3.14), we consider an arc between product i and j, and its self-capacity w_i. If a route follows the path from i to j, and from j to k, then one can impose the following constraints:

$$x_{ij}^{q} + \sum_{\substack{k \in \mathcal{V}' \\ w_i < q_j + q_k}} x_{jk}^{q-q_j} \leq 1 \quad i, j \in \mathcal{V}', q \geq q_j \in \mathcal{C}. \tag{3.14}$$

We can also use the self-capacity to derive a new family of valid inequalities. Let $\mathcal{S}_1$ represent a set of fragile products and its complement denoted $\mathcal{S}_2$, and all arcs going from $i \in \mathcal{S}_1$ to any product j. We know that all arcs going out of j to a product k with the sum of weights $q_k + q_j$ is larger than the maximum self-capacity of the products in $\mathcal{S}_1$ will not be permitted. So all other self-capacity of any products in $\mathcal{S}_1$ are also violated. Thus, valid inequalities (3.15) forbid this situation:

$$\sum_{i \in \mathcal{S}_1} x_{ij}^{q} + \sum_{\substack{k \in \mathcal{S}_2 \\ \max_{i \in \mathcal{S}_1}\{w_i\} < q_k + q_j}} x_{jk}^{q-q_j} \leq 1 \quad j \in \mathcal{V}', \mathcal{S}_1 \subseteq \mathcal{V}' : w_i < Q, \mathcal{S}_2 \subseteq \mathcal{V}', q \geq q_j \in \mathcal{C}. \tag{3.15}$$

3.3.2 Two-index flow formulation

Formulation F1 has the drawback that the number of variables is dependent on the number of q values that can be obtained by different picking routes. We now provide a two-index flow formulation to determine the best picker routes, based on the model described in [Laporte, 1986, Toth and Vigo, 2014] for the asymmetric VRP. We define new binary variables x_{ij} equal to one if arc (i, j) is used, and an integer variable K indicating the number of picking tours required to satisfy all the orders. This model can be stated as follows:

$$\text{(F2) minimize} \sum_{(i.j) \in \mathcal{A}} d_{ij} x_{ij} \tag{3.16}$$

subject to

$$\sum_{j \in \mathcal{V}'} x_{0j} = K \tag{3.17}$$

$$\sum_{i \in \mathcal{V}'} x_{i0} = K \tag{3.18}$$

$$\sum_{i \in \mathcal{V}} x_{ij} = 1 \quad j \in \mathcal{V}' \tag{3.19}$$

$$\sum_{j \in \mathcal{V}} x_{ij} = 1 \quad i \in \mathcal{V}' \tag{3.20}$$

$$\sum_{i,j \in \mathcal{S}} x_{ij} \leq |\mathcal{S}| - r(s) \quad \mathcal{S} \subseteq \mathcal{V}', |\mathcal{S}| \geq 2 \tag{3.21}$$

$$K \in \mathbb{N}_+ \tag{3.22}$$

$$x_{ij} \in \{0, 1\} \quad i, j \in \mathcal{V} \tag{3.23}$$

The objective function (3.16) minimizes the total traveled distance. Constraints (3.17) and (3.18) impose the degree requirements for the I/O point. Equations (3.19) and (3.20) are in-degree and out-degree constraints. They state that each product i must be picked exactly once. Constraints (3.21) correspond to generalized subtour elimination constraints. They simultaneously forbid subtours and ensure the capacity requirements with $r(s) = \left\lceil \frac{\sum_{j \in \mathcal{S}} q_j}{Q} \right\rceil$. Constraints (3.22) and (3.23) impose integrality and binary conditions on the variables.

Similarly to the capacity-indexed formulation, one can add beforehand the *category* constraints when creating the variables. We also remove irrelevant arcs going from non-food to food products. However, unlike formulation F1, formulation F2 cannot handle the *fragility* constraints in a preprocessing phase by eliminating infeasible variables. Thus, we propose a branch-and-cut routine that dynamically adds the fragility constraints defined by (3.24). For a subset of nodes $\mathcal{S}$, we check whether the products related to the potential nodes that may follow node i respect the maximum weight supported by i, w_i. If this condition is not met, we then impose the fragility constraints:

$$\sum_{j \in \mathcal{S}} x_{ij} + \sum_{j,k \in \mathcal{S}} x_{jk} \leq |\mathcal{S}| - 1 \quad \mathcal{S} \subseteq \mathcal{V}' : q_s > w_i, i \in \mathcal{V}' \backslash \mathcal{S}. \tag{3.24}$$

The *weight* constraints (3.13) proposed for formulation F1 can be adapted for F2 by removing the capacity index, which results in inequalities (3.25):

$$\sum_{i,j \in \mathcal{S}} x_{ij} + \sum_{l \in \mathcal{S}} \sum_{h \in \mathcal{V}'_h \backslash \mathcal{S}} x_{lh} \leq |\mathcal{S}| - 1 \quad \mathcal{S} \subseteq \mathcal{V}' : q_s \geq L. \tag{3.25}$$

We have also adapted the two sets of valid inequalities (3.14) and (3.15) from model F1 to model F2. The new valid inequalities (3.26) and (3.27) may be written as follows:

$$x_{ij} + \sum_{\substack{k \in \mathcal{V}' \\ w_i < q_j + q_k}} x_{jk} \leq 1 \quad i, j \in \mathcal{V}' \tag{3.26}$$

$$\sum_{i \in \mathcal{S}_1} x_{ij} + \sum_{\substack{k \in \mathcal{S}_2 \\ \max_{i \in \mathcal{S}_1}\{w_i\} < q_k + q_j}} x_{jk} \leq 1 \quad j \in \mathcal{V}' \mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{V}'. \tag{3.27}$$

3.4 Solution algorithms

We now describe the proposed heuristics and exact algorithms to solve the OPP-WFCC introduced in this chapter. We describe four classical heuristic algorithms and the ALNS metaheuristic in Section 3.4.1. Then, we present the branch-and-cut algorithm and the cutting planes in Section 3.4.2.

3.4.1 Heuristic algorithms

In this section we describe five heuristic algorithms to solve the OPP-WFCC. The first four are extensions of classical and well-known heuristics that we have adapted to solve the OPP-WFCC, and the last one is a metaheuristic designed specifically for our problem. In Section 3.4.1 we describe how we have adapted the S-shape, mid point, and largest gap heuristics of Hall [1993] along with the combined heuristic of Roodbergen and De Koster [2001b] for our problem. These solution methods are modified to handle the precedence constraints considered in the OPP-WFCC. Section 3.4.1 presents the main procedures of the ALNS metaheuristic.

Classical heuristics

Under the S-shape heuristic, the picker will completely traverse any aisle containing at least one item to be picked. Thus, the warehouse is completely traversed, by leaving an aisle and entering the adjacent one, picking products as the picker advances.

In the mid point heuristic we divide the warehouse into two halves. There are picks on the front side, reached by the front aisle, and picks on the back side reached from the back side. The picker only crosses through the first and last aisles of the picking tour. Intermediate aisles are never crossed completely. According to Hall [1993], this method performs better than the S-shape when the number of picks per aisle is small.

In the largest gap heuristic, the picker follows the same idea of the S-shape by visiting all adjacent aisles, but instead of traversing the aisle completely, he enters the aisle up to the item that will leave the largest gap for the next item in that aisle. This heuristic tries to maximize the parts of the aisles that are *not* traversed. When the largest gap is identified, the picker returns and leaves the aisle from the same side (back or front) used to enter it. We proceed by making the first pick as in the original heuristic and by traversing the warehouse following the original rules.

In the combined heuristic, picking routes visit every aisle that contains at least one item. We have adapted the dynamic programming algorithm of Roodbergen and De Koster [2001b] to determine whether it is better to traverse the current aisle in full or to go back, then entering the next aisle from the front or from the rear side. In general, this method is best suited for pickings that follow a TSP-like tour.

The following modifications are required to these four heuristics in order to handle weight, fragility, and category constraints. We start by picking the first available product in the left most aisle as in the original version of the heuristics. Any subsequent product is picked if it respects all the constraints of the problem, and skipped otherwise. We continue following the heuristic rules traversing the warehousing, and skipping any infeasible pick until the pallet is full or the last item is reached. At this point, the picker returns to the I/O point and starts a new route going back to the first skipped item, which will be picked and the same procedure is repeated as long as there are products to be picked.

Adaptive large neighborhood search heuristic

We present an implementation of an Adaptive large neighborhood search heuristic (ALNS) metaheuristic, widely based on Ropke and Pisinger [2006]. The ALNS is composed of a set of simple destruction and reconstruction heuristics in order to find better solutions at each iteration using an adaptive layer to keep track of the performance of the invoked heuristics. An initial solution can be considered to speed up the search and the convergence of the algorithm. We have implemented a fast sequential insertion heuristic, which performs a greedy search for the best insertion for one product at a time.

The ALNS selects one of many destroy and repair operators at each iteration. We have implemented three destroy and two repair operators. Destroy operators include the Shaw removal [Shaw, 1997], the worst removal, and a random removal. Repair operators include a greedy parallel insertion and a k-regret heuristic [Potvin and Rousseau, 1993]. Each operator is selected with a probability that depends on its past performance, and a simulated annealing acceptance criterion is used. Each procedure of the ALNS is adapted to deal with the weight, fragility, and category constraints.

The ALNS is run for 50,000 iterations of destroy-repair operations. After every 100 iterations the weight of each operator is updated according to its past performance. Initially, all the operators have the same weight. A sketch of our ALNS implementation is provided in Algorithm 4 (for further details see Ropke and Pisinger [2006]).

3.4.2 Exact algorithms

In this section we present the various algorithms used to solve exactly the mathematical models from Section 3.3. We present in Section 3.4.2 a subset sum algorithm that enables

Algorithm 4 Adaptive large neighborhood search metaheuristic

```
Create an initial solution s using sequential insertion operators : s' = s
D : set of removal operators, I : set of repair operators
Initiate probability ρ^d for each destroy operator d ∈ D and probability ρ^i for each repair operator i ∈ I
while stopping criterion is not met do
    Select remove (d ∈ D) and insert (i ∈ I) operators using ρ^d and ρ^i
    Apply operators and create s^t
    if s^t, is accepted by the simulated annealing criterion then
        s = s^t
    end if
    if cost(s) < cost(s') then
        s' = s
    end if
    Update ρ^d and ρ^i
end while
return s'
```

us to drastically reduce the number of required variables for model F1. This is followed in Section 3.4.2 by an overview of the branch-and-cut algorithm we have used as well as the detailed description of the procedures used to dynamically identify and generate violated valid inequalities.

Subset sum algorithm to reduce the number of variables of model F1

The variables of formulation F1 presented in Section 3.3.1 can only be fully enumerated for small and medium size instances. However, it is easy to observe that some variables are never used in the model, e.g., the ones for which some values of q cannot be obtained by any combination of the weights of the products. These variables can be generated and fed into the solver, which will set them to zero in any feasible solution. If one can identify these variables beforehand, it is possible to set them to zero and remove them from the model at a preprocessing phase. Thus, one can (substantially) decrease the size of the model and the memory footprint by preprocessing the model and the instance a priori, identifying the subset of variables that should not be generated.

We use a subset sum algorithm to identify all possible values of q from 1 to Q that can be achieved by any combination of weights q_i. The ones that are found not to be feasible are not generated.

From a theoretical point of view, its performance is directly related to the distribution of the weights of the items in the instance. For example, an instance for which all products have a weight of five units will have five time less variables than the model with all the values of q.

Branch-and-cut algorithms

Models F1 and F2 can be fed straightforwardly into a general purpose solver and solutions are obtained by branch-and-bound if the number of constraints (3.13), (3.21), (3.24) and (3.25) is not excessive. However, for instances of realistic size, the number of these constraints is too large to allow a full enumeration and they must be dynamically generated throughout the search process. Indeed, polynomial constraints may be added a priori to the model while other constraints cannot be generated a priori since their number is exponential.

The exact algorithm we present is a branch-and-cut scheme in which inequalities (3.13), (3.21), (3.24), and (3.25) are generated and added into the program whenever they are found to be violated. It works as follows. At a generic node of the search tree, a linear program containing the model with a subset of the subtour elimination constraints and relaxed integrality constraints is solved, a search for violated inequalities is performed, and some of these are added to the current program which is then reoptimized. This process is reiterated until a feasible or dominated solution is reached, or until there are no more cuts to be added. At this point, branching on a fractional variable occurs. We provide a sketch of the branch-and-cut scheme in Algorithm 5. Note that for formulation F1, we apply Algorithm 5 but we skip the process to identify violated subtour elimination constraints (lines $8-12$) as connectivity requirements are ensured by constraints (3.7).

Algorithm 5 Branch-and-cut algorithm

```
1: Subproblem solution: Solve the LP relaxation of the current node
2: Termination check:
3: if there are no more nodes to evaluate then
4:    Stop
5: else
6:    Select one node from the branch-and-cut tree
7: end if
8: while the solution of the current LP relaxation contains subtours do
9:    Identify connected components as in Padberg and Rinaldi [1991]
10:   Add violated subtour elimination constraints
11:   Subproblem solution. Solve the LP relaxation of the current node
12: end while
13: if the solution contains no disconnected components then
14:   Apply Algorithms 6 and 7, and add violated cuts
15: end if
16: if the solution of the current LP relaxation is integer then
17:   Go to the termination check
18: else
19:   Branching: branch on one of the fractional variables
20:   Go to the termination check
21: end if
```

In this branch-and-cut algorithm, weight, and fragility inequalities are used as cutting planes

to strengthen the linear programming relaxation at each node of the branch-and-bound tree. Constraints (3.13), (3.24), and (3.25) cannot be generated a priori since their number is exponential. These are initially relaxed and dynamically generated as cuts as they are found to be violated.

When model F1 without *weight* constraints (3.13) (similarly with (3.25) for F2) is solved, two situations can occur. The first one consists of finding an integer solution. Then one can easily verify if the picking tour exceeds the weight limit L or the self-capacity w_i by calculating the cumulative weight of a set $\mathcal{S}$. In the second case, when the solution is fractional, one can identify connected components by means of the maximum flow algorithm as in Padberg and Rinaldi [1991]. This procedure, sketched in Algorithm 6, consists of constructing an auxiliary graph as follows. First, a node i is selected. Then the node j associated with the maximum flow value leaving i is identified and added to the set $\mathcal{S}$. We check whether the sum of weights of the products included in $\mathcal{S}$, referred to by $q_{\mathcal{S}}$, respects the threshold L. If it exceeds this limit, we add cuts to forbid such solution. This is achieved by adding the appropriate constraints (3.13) for the capacity indexed formulation, and (3.25) for the two-index flow formulation associated with the nodes in $\mathcal{S}$.

Similarly, in Algorithm 7, we describe the procedure used to dynamically generate constraints (3.24). We identify a subset $\mathcal{S}$ such that it respects the *fragility* constraints. Otherwise, we add the violated constraints associated with the nodes in $\mathcal{S}$.

Algorithm 6 *Weight* constraints algorithm

1: **for** $i = 1$ to m **do**
2: $\mathcal{S} = \{i\}$
3: $j^* = argmax_{k \in \mathcal{V}' \setminus \mathcal{S}}\{x_{ik}\}$
4: $\mathcal{S} = \mathcal{S} \cup \{j^*\}$
5: **if** $q_{\mathcal{S}} > L$ and $q_{\mathcal{S}} \leq Q$ **then**
6: **for** $l \in \mathcal{S}$ **do**
7: **if** the solution violates (3.13) (or (3.25)) **then**
8: Add *weight* constraints (3.13) (or (3.25))
9: **end if**
10: **end for**
11: **end if**
12: **if** $q_{\mathcal{S}} > Q$ **then**
13: Continue to next i from step 1
14: **else**
15: Go to Step 3
16: **end if**
17: **end for**

Algorithm 7 *Fragility* constraints algorithm

for $i = 1$ to m **do**
  if i is a fragile product **then**
    $\mathcal{S} = \{i\}$, best $= i$
    while stop criterion is not met **do**
      $j^* = argmax_{k \in \mathcal{V}' \setminus \mathcal{S}} \{x_{best,k})\}$
      if $x_{best,k} < 0.5$ **then**
        Stop
      end if
      $\mathcal{S} = \mathcal{S} \cup \{j^*\}$, Best$=j$
      if $q_{\mathcal{S}} \leq Q$ and $q_{\mathcal{S}} > w_i$ **then**
        if the solution violates (3.24) **then**
          Add *fragility* constraints (3.24)
        end if
      end if
      if $q_{\mathcal{S}} > Q$ **then**
        Stop
      end if
    end while
  end if
end for

3.5 Computational experiments

In this section, we provide details on the implementation, benchmark instances, and describe the results of extensive computational experiments. Implementation and hardware information is provided in Section 3.5.1. The description of the benchmark instances is presented in Section 3.5.2, followed by the results of our computational experiments. Heuristic results are presented in Section 3.5.3 and the results of experiments carried out in order to assess the performance of the proposed exact methods are presented in Section 3.5.4.

3.5.1 Implementation details

All the formulations described in Section 3.3 and the algorithms described in Section 3.4 were implemented in C++. The branch-and-cut algorithm uses the CVRPSEP library [Lysgaard, 2003] for the sub-tours elimination constraints, the newly proposed cutting methods, and IBM CPLEX Concert Technology 12.6 as the branch-and-bound solver. All computations were executed on machines equipped with two Intel Westmere EP X5650 six-core processors running at 2.667 GHz, and with up to 8 GB of RAM installed per node running the Scientific Linux 6.3. All algorithms were provided a time limit of 7200 seconds.

3.5.2 Data sets generation

Since no data sets are available for the OPP-WFCC, we have created three groups of instances to represent different configurations and combinations of the new features introduced in this chapter. To reflect common practice, non-food items are grouped in the same aisle(s). In the first group of instances, called G1, only one side of the first aisle is dedicated to non-food items, whereas all fragile and non-fragile products are randomly placed throughout the warehouse. In the second group of instances (G2), we split the picking area in two symmetric zones. This allows non-food items to be placed in the lateral extremities of the warehouse. All remaining items are placed randomly elsewhere. In the third group (G3), non-food items are placed in the extremities like in G2, but solid products (SP), i.e., non-fragile, with large self-capacity are grouped together in the sections close to the I/O point. These three types of layouts are illustrated in Figure 3.4.

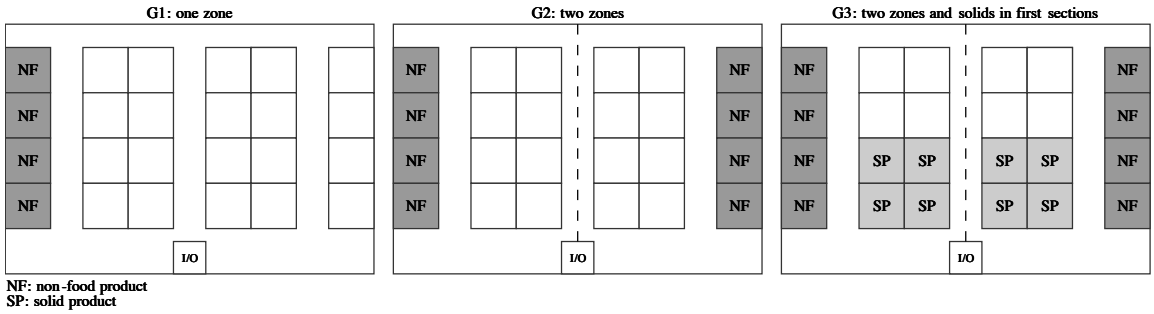


Figure 3.4 – Schematic representation of the three groups of instances

In each group of instances, we have kept the same proportion of food and non-food products, and ranges of weight and self-capacity. The generator has a 50% probability to create a fragile product. Each fragile product has a 2/3 probability to be a food product and 1/3 to be non-food. A non-fragile product, food or otherwise, has a weight between five and 50 kg, and infinite self-capacity. A fragile product has a weight between one and 10 kg, and a self-capacity between five and 50 kg.

The general characteristics of this large test bed are summarized in Table 3.1. An instance is characterized by the number of picks, the number of aisles, and the capacity of the picking vehicles. The number of SKU to be picked is between 20 and 100, by steps of 10. The number of aisles is equal to 3, 4, 5, 6 or 7, with the I/O point located in the middle of the warehouse. The capacity of the pallet may take the value 150 or 250. In total, there are 270 different parameter combinations represented in Table 3.1. For each combination, we have generated randomly three instances, for test bed of 810 distinct instances. Recall that each aisle side contains 20 storage locations and the number of SKU per warehouse depends on the number of aisles. Regarding the parameters values, we have used the following physical distance parameters: $\alpha = 10$, $\beta = 5$, $\gamma = 15$, and $\Omega = 5$.

Table 3.1 – General characteristics of the generated test bed

Parameter	Variations	Values
group	3	G1, G2, G3
# of picks	9	20, 30, 40, 50, 60, 70, 80, 90, 100
# of aisles	5	3, 4, 5, 6, 7
capacity of the pallet	2	150, 250

3.5.3 Heuristic results

This section presents the results of extensive computational experiments of the five heuristic algorithms presented in Section 3.4.1 for the test bed containing 810 instances. Table 3.2 presents the solutions of the heuristic algorithms for all the three groups of instances. Specifically, for each heuristic, group, and pallet capacity, and for each number of pickups we provide the total distance which is the average over 15 instances, i.e., three instances for each scenario. We also provide the average distance over each group.

The best average results of classical heuristics are underlined for each capacity and each group. In terms of computational times required by the different heuristic algorithms, the CPU time needed is less than one second for instances with 100 pickups and seven aisles. The classical heuristics results presented in Table 3.2 highlight that their performance slightly differ over the three instances groups. The mid-point heuristic is most likely less effective than the other heuristics. For the first group, the combined heuristic outperforms the other three. For G2, the S-shape heuristic is the best one. Finally, for G3, the largest gap heuristic yields the best results. These results hold for both capacity scenarios. More generally, the four classical heuristics yielded very similar solutions, whereas these may be significantly enhanced by the ALNS metaheuristic.

A richer and more intricate metaheuristic, such as the ALNS, significantly improves the solution, as shown in Table 3.2. The average solution is reduced by almost 50% for all groups. This may be explained by the fact that the ALNS metaheuristic provides more compact solutions with fewer picking tours and much shorter distances, thanks to its intensification and diversification procedures. Regarding the running time, the ALNS improves the solution quality with no significant time increase. The average running time is around 60 seconds for the largest instances with 100 pickups and 7 aisles. As expected, increasing the capacity of the pallet (as done in the second scenario) reduces the distance traversed, and this for all heuristics. Moreover, being the most advanced one, the ALNS better exploits this extra capacity and reduces distances significantly.

Table 3.2 – Average heuristic results on instances with $Q = 150$ and $Q = 250$

	# pickups	$Q = 150$ S-shape	Mid point	L-Gap	Combined	ALNS	$Q = 250$ S-shape	Mid point	L-Gap	Combined	ALNS
G1	20	2169.00	2224.00	2205.67	2016.67	1234.33	1917.00	2036.00	1826.33	1852.00	1082.33
	30	2949.00	3014.33	2902.33	2850.00	1555.33	2638.67	2891.67	2715.67	2443.67	1357.67
	40	3665.33	3812.33	3558.33	3611.67	1904.67	3075.67	3281.67	3145.33	2967.00	1552.67
	50	4075.67	4223.33	4126.00	3972.33	2135.33	3754.00	3917.67	3892.00	3562.33	1831.33
	60	5181.33	5114.33	4912.67	5023.33	2540.33	4357.67	4311.00	4240.67	4157.00	2063.33
	70	5694.67	5680.00	5409.00	5600.67	2739.33	4816.33	4935.33	4788.33	4615.00	2259.33
	80	6159.33	6135.00	6012.33	5750.33	3046.33	5329.00	5226.67	5215.67	5029.33	2382.33
	90	7011.67	6878.00	6886.67	6877.00	3408.33	5651.67	5919.67	5579.00	5331.00	2580.00
	100	7663.67	7550.00	7452.67	7147.00	3716.67	5981.00	6084.33	6045.67	5897.33	2738.00
	Average	4952.19	4959.04	4829.52	4761.00	2475.63	4169.00	4289.33	4160.96	3983.85	1983.00
G2	20	2231.00	2441.00	2288.00	2213.00	1290.67	1959.33	2171.33	2077.00	1961.33	1171.00
	30	3093.33	3274.67	3154.67	2996.00	1641.33	2651.33	2950.00	2835.00	2747.00	1450.00
	40	3695.33	3757.67	3765.33	3728.33	1967.00	3398.33	3514.33	3527.00	3263.67	1693.67
	50	4150.00	4303.00	4295.33	4244.33	2219.67	3878.00	4322.33	3961.33	3814.67	1883.67
	60	5034.33	5403.00	5044.00	5249.33	2568.00	4445.67	4469.33	4487.00	4473.33	2135.33
	70	5608.67	5874.67	5714.67	5648.00	2797.00	4667.33	5089.33	4858.67	4993.33	2351.00
	80	6048.00	6376.33	6132.67	6142.67	3133.33	5201.00	5223.00	5250.67	5218.33	2446.67
	90	6921.67	7087.67	6973.67	6719.00	3466.33	5448.67	5808.00	5469.00	5519.00	2665.33
	100	7453.67	7680.67	7338.00	7625.67	3773.67	6180.67	6239.00	6042.67	6074.00	2764.44
	Average	4915.11	5133.19	4967.37	4951.81	2539.67	4203.37	4420.74	4278.70	4229.41	2062.35
G3	20	2311.00	2438.33	2205.67	2199.67	1279.67	2047.67	2147.33	2003.33	2025.67	1195.00
	30	2977.00	3183.00	3071.33	2982.67	1620.67	2686.00	2990.00	2695.00	2742.00	1439.00
	40	3590.00	3714.00	3455.67	3732.00	1919.00	3370.33	3484.67	3356.00	3230.67	1676.67
	50	4173.67	4400.00	4110.67	4248.67	2158.33	3736.33	4113.33	3758.33	3908.33	1862.33
	60	4911.00	5084.33	4980.33	5099.67	2487.33	4387.33	4501.33	4325.67	4455.00	2102.67
	70	5415.33	5651.00	5428.00	5779.00	2707.33	4660.00	5265.00	4734.00	4980.33	2328.67
	80	5949.00	6030.67	5904.67	6111.67	3011.67	5410.33	5243.00	5113.00	5224.33	2377.00
	90	6892.33	6828.33	6857.67	6738.00	3327.00	5371.67	5764.33	5353.00	5568.00	2606.33
	100	7262.67	7380.33	7132.67	7786.67	3586.67	5959.33	6125.33	5826.33	6023.33	2764.67
	Average	4831.33	4967.78	4794.07	4964.22	2455.30	4181.00	4403.81	4129.41	4239.74	2039.15
Overall average		4899.54	5020.00	4863.65	4892.35	2490.20	4184.46	4371.30	4189.69	4151.00	2028.16

3.5.4 Exact algorithms results

Tables 3.3 and 3.4 present the results of the best heuristic proposed (ALNS) compared to the results obtained by solving the two mathematical models, F1 and F2, when applying the branch-and-cut algorithms with the proposed cutting planes.

For each instance group and for each number of pickups, we provide the average results over all the instances and over all the number of aisles, with $Q = 150$ in Table 3.3 and $Q = 250$ in Table 3.4. We report the upper bound (UB) and the lower bound (LB) for models F1 and F2. We provide the average percentage gaps, given by the ratio ($\frac{UB-LB}{LB} \cdot 100$). We also provide the average running time in seconds.

From Tables 3.3 and 3.4, we observe that the results provided by the ALNS metaheuristic are close to the best UBs yielded by both models. For instance, when $Q = 150$, ALNS gives an overall average distance of 2496.83 compared to 2482.68 and 2479.07 for F1 and F2. This remark also holds when $Q = 250$. Moreover, the ALNS algorithm provides near-optimal solutions within very short computing times (few seconds) compared to the time spent by both models to prove optimality or to provide better UBs. However, deriving better results at the expense of longer run times was one of our goals to provide the best possible results for the newly proposed testbed and serve the warehousing research community.

A deeper analysis of the formulations shows that the solutions are tight, and the optimality gap is in average 9.35% when $Q = 150$ for F1, and 6.91% for F2. Model F2 proves optimality over all instances with 20 pickups and almost all instances with 30 pickups within very short computing times. Both models had problems closing the gap and proving optimality for instances with more than 30 pickups.

Note that the time limit of 7200 seconds is often exceeded by F1. This is due to the fact that the time spent to instantiate model F1 is also considered in the total time, and the size of model F1 may become an issue when the instance size increases. For example, for instances including 100 pickups, almost two hours are needed only to create the model.

Finally, it is important to notice that the subset sum algorithm presented in Section 3.4.2 was able to reduce an average of 12.55% variables compared to a complete enumeration of all possible variables for model F1. In our test bed, we have observed a reduction of up to 23.56% in the number of variables, and a minimum reduction of 5.87%.

Regarding the three groups of instances, we observe that the gap between the upper and lower bounds is slightly lower for group G1 for both models. There are no major differences among groups in terms of the traveled distance except for very small variations. Indeed, it seems more difficult to solve instances where the products are placed in groups, in such a way as to facilitate the picking work.

Table 3.3 – Average results of the exact algorithms per group of instances and number of pickups on instances with $Q = 150$

Group	Pickups	ALNS	Time (s)	Capacity indexed formulation (F1)				Two-index formulation (F2)			
				UB	LB	%Gap	Time (s)	UB	LB	%Gap	Time (s)
G1	20	1251.00	1.53	1232.00	1232.00	0.00	118.27	1232.00	1232.00	0.00	8.00
	30	1568.00	3.33	1542.67	1521.86	1.37	1577.60	1542.67	1540.30	0.15	541.60
	40	1928.00	5.53	1892.67	1798.71	5.22	7278.67	1888.67	1873.72	0.80	1656.40
	50	2127.33	9.60	2120.67	1937.19	9.47	7604.07	2104.33	2031.58	3.58	5321.40
	60	2530.33	16.40	2529.33	2319.77	9.03	8678.60	2525.67	2380.87	6.08	7200.67
	70	2734.00	24.47	2733.33	2491.91	9.69	9527.27	2732.33	2595.60	5.27	7201.00
	80	3040.67	31.20	3038.33	2790.07	8.90	10031.27	3035.00	2854.34	6.33	7201.20
	90	3403.33	50.33	3402.67	3083.69	10.34	16523.53	3403.33	3153.14	7.93	7202.67
	100	3722.67	65.60	3722.67	3373.26	10.36	19710.80	3722.67	3426.66	8.64	7202.93
	Average	2478.37	23.11	2468.26	2283.16	8.11	9005.56	2465.19	2343.13	5.21	4837.32
G2	20	1300.33	1.07	1287.67	1252.50	2.81	188.53	1287.67	1287.67	0.00	99.73
	30	1661.67	3.20	1627.67	1601.29	1.65	3185.60	1627.00	1626.52	0.03	1066.07
	40	2012.33	6.13	1962.67	1820.88	7.79	7140.73	1955.67	1890.01	3.47	4403.53
	50	2230.00	9.40	2221.33	1993.66	11.42	7946.80	2193.00	2079.29	5.47	6028.47
	60	2565.33	17.13	2557.00	2301.68	11.09	8804.27	2546.00	2371.23	7.37	6923.13
	70	2803.33	24.80	2797.33	2519.51	11.03	9967.53	2798.00	2574.98	8.66	7201.13
	80	3129.67	33.73	3129.33	2805.88	11.53	11105.93	3113.00	2848.36	9.29	7201.60
	90	3453.67	51.00	3453.67	3116.54	10.82	14786.20	3453.67	3165.65	9.10	7202.33
	100	3783.00	67.53	3781.00	3392.42	11.45	17634.47	3783.00	3420.28	10.60	7202.93
	Average	2548.81	23.78	2535.30	2311.60	9.68	8973.34	2528.56	2362.66	7.02	5258.77
G3	20	1275.33	1.67	1253.00	1140.00	9.91	211.47	1253.00	1253.00	0.00	7.67
	30	1614.67	3.13	1592.33	1592.33	0.00	1559.87	1592.33	1586.33	0.38	850.07
	40	1958.33	5.60	1871.67	1781.19	5.08	6868.73	1879.33	1816.53	3.46	3475.00
	50	2139.67	8.87	2119.00	1939.38	9.26	7849.73	2109.33	1995.65	5.70	6013.20
	60	2490.67	16.27	2474.00	2206.25	12.14	8744.40	2473.00	2254.63	9.69	6729.33
	70	2706.67	20.80	2706.67	2431.64	11.31	9096.13	2702.67	2448.13	10.40	7201.00
	80	3042.33	31.00	3041.67	2678.45	13.56	11614.67	3041.67	2679.62	13.51	7201.80
	90	3360.67	43.13	3360.67	2991.94	12.32	12734.93	3358.67	3029.84	10.85	7201.80
	100	3581.33	66.87	3581.33	3188.99	12.30	16577.47	3581.33	3195.08	12.09	7202.13
	Average	2463.30	21.93	2444.48	2216.69	10.28	8361.93	2443.48	2250.98	8.55	5098.00
Overall average		2496.83	22.94	2482.68	2270.48	9.35	8780.28	2479.07	2318.93	6.91	5064.70

Table 3.5 presents the number of optimal solutions obtained by both models over the 810 instances. These results are separated in groups of instances, number of aisles, and by the value of the capacity Q. This helps highlight the effect of each of these characteristics. In the first and second lines we present the capacity indexed formulation F1 and the two-index formulation F2 with the two capacities, $Q = 150$ and $Q = 250$. We then present the results for each group of instances and each number of aisles. The number of optimal solutions is slightly higher with F2. This corroborates the results already observed from the gaps and running time of these models. A transversal analysis of Tables 3.3, 3.4, and 3.5 points out again the difficulty of the problem. We observe that the best exact algorithm is able to prove optimality for only 27% of the instances.

Table 3.6 presents a deeper statistical comparison between the proposed formulations. For each model, we provide averages for the percentage optimality gap, the computation time in seconds, the number of variables, and constraints generated along with the number of cuts added to fractional and integer solutions, and finally the number of nodes explored in the branch-and-bound tree. The last column of the table shows the relative difference of F1 with respect to F2.

The first outstanding result from Table 3.6 is related to the huge number of variables and

Table 3.4 – Average results of the exact algorithms per group of instances and number of pickups on instances with $Q = 250$

Group	Pickups	ALNS	Time (s)	Capacity indexed formulation (F1)				Two-index formulation (F2)			
				UB	LB	%Gap	Time (s)	UB	LB	%Gap	Time (s)
G1	20	1093.67	1.73	1064.33	1048.65	1.50	1644.87	1063.33	1063.33	0.00	1.07
	30	1388.00	4.40	1360.33	1314.20	3.51	4645.33	1355.67	1355.67	0.00	45.80
	40	1568.67	6.87	1559.33	1422.70	9.60	7523.00	1530.33	1517.27	0.86	2241.00
	50	1859.33	12.07	1858.67	1584.14	17.33	8551.40	1836.00	1701.57	7.90	7081.73
	60	2057.67	20.33	2057.00	1746.16	17.80	12133.93	2016.33	1847.55	9.14	7065.67
	70	2236.67	31.80	2236.67	1911.78	16.99	14281.20	2228.00	1987.49	12.10	7200.93
	80	2387.00	46.07	2387.00	1999.96	19.35	17920.47	2377.33	2043.08	16.36	7201.13
	90	2568.00	73.67	2568.00	2157.94	19.00	35708.87	2567.33	2214.82	15.92	7202.07
	100	2716.67	86.27	2716.67	2283.53	18.97	41148.93	2716.00	2323.99	16.87	7201.73
	Average	1986.19	31.47	1978.67	1718.78	15.12	15950.89	1965.59	1783.86	10.19	5026.79
G2	20	1209.33	1.67	1164.67	1149.44	1.32	2309.47	1164.67	1164.67	0.00	0.33
	30	1499.67	3.87	1470.33	1357.62	8.30	5233.27	1448.00	1444.69	0.23	529.07
	40	1728.67	7.40	1714.67	1526.73	12.31	8008.13	1699.00	1641.20	3.52	4543.80
	50	1935.67	12.07	1935.67	1613.44	19.97	9346.67	1891.33	1673.86	12.99	6787.13
	60	2179.67	22.80	2179.67	1810.35	20.40	12049.20	2141.67	1876.63	14.12	6893.53
	70	2379.67	31.20	2377.67	1973.70	20.47	15551.13	2354.00	2005.31	17.39	7201.27
	80	2473.00	46.27	2473.00	2037.14	21.40	21498.93	2473.00	2072.88	19.30	7201.53
	90	2694.67	65.47	2694.67	2230.08	20.83	33595.00	2694.00	2266.36	18.87	7202.27
	100	2825.33	80.07	2825.33	2351.37	20.16	40648.80	2823.33	2356.54	19.81	7202.47
	Average	2102.85	30.09	2092.85	1783.32	17.36	16471.18	2076.56	1833.57	13.25	5284.60
G3	20	1235.33	1.53	1193.33	1177.85	1.31	2571.73	1193.00	1193.00	0.00	0.60
	30	1481.00	4.00	1452.00	1357.24	6.98	4982.13	1441.00	1441.00	0.00	178.53
	40	1730.00	6.87	1720.00	1508.52	14.02	7646.07	1676.00	1633.55	2.60	3854.47
	50	1876.00	13.27	1876.00	1602.06	17.10	8903.73	1856.67	1680.63	10.47	6233.33
	60	2149.00	20.27	2149.00	1795.72	19.67	11513.80	2106.33	1817.12	15.92	7200.73
	70	2321.67	28.13	2321.67	1938.48	19.77	14725.33	2304.33	1962.90	17.39	7201.07
	80	2384.67	44.53	2384.67	2008.36	18.74	22755.27	2383.33	2007.33	18.73	7201.67
	90	2641.00	55.67	2641.00	2179.95	21.15	28728.27	2640.33	2202.66	19.87	7201.73
	100	2780.67	80.00	2780.00	2311.15	20.29	35836.53	2778.67	2291.00	21.29	7201.60
	Average	2066.59	28.25	2057.52	1764.37	16.61	15295.87	2042.19	1803.24	13.25	5141.53
Overall average		2051.88	29.94	2043.01	1755.49	16.38	15905.98	2028.11	1806.89	12.24	5150.97

Table 3.5 – Number of optimal solutions per group of instances and capacity of the truck

	Capacity	G1						G2						G3						Total
		3	4	5	6	7	Total	3	4	5	6	7	Total	3	4	5	6	7	Total	
Q=150	F1	6	7	5	6	6	30	4	5	4	6	6	25	8	7	6	7	5	33	88
	F2	8	9	10	11	8	46	8	7	9	8	9	41	8	8	9	9	7	41	128
Q=250	F1	5	4	4	4	6	23	4	4	4	3	5	20	4	4	3	4	5	20	63
	F2	9	9	8	9	9	44	6	7	6	8	10	37	7	8	7	9	10	41	122

constraints generated in F1. These two figures are more than 100 times higher than the number of variables and constraints generated in F2. Although the subset sum algorithm helps decreasing the size of F1 by removing irrelevant variables, the size of formulation F1 remains huge compared to F2 and requires much more time to load the instance and create the model.

However, F1 uses almost no cuts during the search process to limit the size of the solution while F2 invokes a huge number of cuts for both fractional and integer solutions. Indeed, over all instances, F2 uses on average 316.2 cuts on fractional solutions and 9703.5 on integer ones. This is due to the fact that F1 does not need to add sub-tours elimination constraints or self-capacity constraints during the search process. Moreover, for both models and on instances with more than 40 pickups, the number of visited nodes gradually decreases while the instances size increases. This is due to the increasing number of cuts generated while the

instances size increases and then a smaller tree size is explored. This shows that the average number of visited nodes in F1 is less than the number of visited nodes in F2 by 33%. These results clearly show that formulation F1 is again outperformed by formulation F2.

Table 3.6 – Statistical comparison of models F1 and F2 over all instances per number of pickups

	Pickups	20	30	40	50	60	70	80	90	100	Average	vs. F2
F1	% Gap	3.2	4.1	9.2	14.5	15.5	15.0	15.7	15.8	15.6	12.0	42%
	Time (s)	1174.1	3530.6	7410.9	8367.1	10320.7	12191.4	15821.1	23679.5	28592.8	12343.1	142%
	# Variables	37479.6	72740.4	130246.9	203011.6	280638.7	384998.3	497471.6	665514.9	793555.8	340628.6	10528%
	# Constraintes	12962.8	23821.5	37205.6	57058.8	75440.1	98905.7	130409.9	159564.4	194357.4	87747.3	11308%
	Fractional cuts	0.6	1.1	1.4	0.2	0.1	0.2	0.1	0.0	0.0	0.4	-100%
	Integer cuts	12.6	17.2	23.2	10.9	7.3	5.5	6.7	2.3	3.3	9.9	-100%
	# Nodes	12998.2	32487.2	31799.1	13686.6	4279.6	2436.6	1504.5	373.6	320.4	11098.4	-33%
F2	% Gap	0.0	0.2	2.7	8.0	10.9	12.0	14.1	13.8	14.9	8.5	
	Time (s)	19.6	535.2	3362.4	6244.2	7002.2	7201.1	7201.5	7202.1	7202.3	5107.8	
	# Variables	330.9	703.7	1218.8	1895.9	2698.1	3691.6	4748.4	6121.6	7435.2	3204.9	
	# Constraintes	101.5	198.8	312.9	486.3	671.4	880.7	1163.6	1378.6	1728.5	769.2	
	Fractional cuts	96.7	277.4	455.4	581.3	476.9	283.7	273.4	206.9	193.9	316.2	
	Integer cuts	128.5	1144.2	5485.5	10890.1	14157.9	14544.8	14606.4	13246.6	13127.8	9703.5	
	# Nodes	1838.3	19307.3	35759.4	34536.0	19465.7	14024.2	10649.1	6882.4	5773.8	16470,7	

3.6 Conclusions

We have tackled a real-world-based and rich order picking problem arising in the grocery retail industry which, to our knowledge, was studied here for the first time. This practical problem extended classical warehousing problems by incorporating more challenges regarding the physical characteristics of the products to be picked. Specifically, products can support a maximum weight when being transported on a lift-truck used for warehouse picking, some products are more fragile than others, and products belonging to food and non-food categories must be picked in a given order such as to avoid contamination. We have adapted four classical order picking heuristics from the warehousing literature to handle these new features. Notably, we have shown how the S-shape, the largest gap, the mid point, and the combined heuristics can yield feasible solutions within very short computation times. Moreover, we have also proposed a more powerful metaheuristic, ALNS, which outperforms the best results obtained by the four heuristics. We have presented two mathematical models including known and new valid inequalities for this challenging picking problem. The first one is the capacity-indexed formulation, which is a based compact single commodity flow using binary variables to indicate the flow on each arc. The second formulation is a two-index flow formulation, in which individual hand-lift trucks are not explicitly identified. We have proposed exact algorithms for their resolution.

We have tested the proposed heuristics, metaheuristic, and the two formulations on large sets of newly generated and realistic instances. The results show the effectiveness of the proposed heuristics in finding high-quality solutions within a negligible computation time. Solutions provided by the ALNS metaheuristic dominated those from the other heuristics, as it traversed smaller distances due to fewer tours. Moreover, we have been able to prove optimality for several instances and to obtain better solutions and tight gaps with two mathematical models that were solved with classical and ad-hoc valid inequalities and cutting planes. Ex-

tensive tests have shown that the proposed exact algorithms outperformed the solutions of the heuristic algorithms. Moreover, we have shown that the two-index formulation outperforms the capacity-indexed formulation for this problem. The most remarkable conclusion for this work is that the running times of the heuristics are very low, and those of the exact algorithms are still acceptable, allowing for their usage in practical applications.

Our future works aim to show how the proposed solution methods can be adapted to cover a variety of other warehousing applications, in other industries and under different assumptions. In addition, there might be a strong correlation between the instances characteristics, e.g., warehouse design, number of aisles, capacity of the truck, and the results obtained. Consequently, more research is needed to demonstrate such interactions through design experiments, as they are supposed to significantly help find the best warehouse design with respect to the routing policy under different conditions. Similar work has been already performed in Chackelson et al. [2013]. Future research that supports this study may include more computational experiments to assess the impact of each precedence constraint on the performance of the heuristic and exact algorithms. Previous studies evaluating the impact of different operating conditions on the route distance exist, e.g., Petersen [1997], Petersen [1999].

Chapter 4

The warehouse products reassignment problem

Résumé

Depuis plusieurs décennies, les chercheurs ont développé des techniques d'optimisation pour les opérations d'entreposage. Ces techniques sont liées aux stratégies de manutention, de préparation de commandes et de stockage pour une myriade de configurations d'entrepôt. On néglige souvent la mise à jour de ces stratégies afin de s'adapter aux changements des offres de produits. La plupart des recherches sur le positionnement des produits fournissent des méthodes pour déterminer où les produits doivent être localisés. Cependant, la partie manutention du problème est souvent mise de côté. Changer la configuration nécessite beaucoup de travail et perturbe les opérations régulières de prélèvements. Ce chapitre présente le problème de réaffectation afin de minimiser la charge de travail. Nous démontrons comment on peut passer d'une affectation de stockage périmée à une meilleure, dans un minimum de temps. Nous présentons trois formulations mathématiques différentes et les comparons à l'aide d'expérimentations approfondies afin d'identifier la meilleure.

Chapter information A research paper based on this chapter, named *Mathematical models for the warehouse reassignment problem*, has been submitted to the journal Computers & Operations Research by Chabot T., Coelho L. C. and Renaud J. in February 2018.

Abstract

The last part of this thesis considers another type of material handling within the picking area: how to move products to new locations. For several decades, researchers have developed optimization techniques for warehouse operations. These techniques are related in particular to the material handling, the order picking and storage assignment strategies for a myriad of warehouse configurations. It is often neglected that these strategies need to be regularly adjusted in order to adapt to changes in the demand and/or product offers. For example, results from Chapter 3 could be quite different with an updated assignment of products, corresponding to the order previsions. Most research on storage assignment provide excellent methods to determine where products should be located. However, the handling part of the problem is often set aside. Moving from one setup to another requires a large amount of work and disturbs regular order-picking operations. This chapter presents the warehouse reassignment problem in order to minimize the total workload to reassign the products to their new locations. We demonstrate how one can move from an out-of-date storage assignment to a better one, in a minimum of working time. We introduce three different mathematical formulations and compare them through extensive computational experiments in order to identify the best one.

4.1 Introduction

Warehouse operations are critical for the performance of distribution centers (DCs) and to the efficiency of supply chains. Placing products in the warehouse and picking them later are some of the most time and cost-consuming activities [De Koster et al., 2007, Gu et al., 2007, Chiang et al., 2011]. A good product location is crucial given the ever-increasing number of products and the pressure for shorter lead times [De Koster et al., 2007, Hong et al., 2012, Tompkins et al., 2010]. Determining the best assignment of items to locations is known in the literature as the storage location assignment problem [Hausman et al., 1976]. A storage assignment strategy is a set of rules which can be used to determine the best place to store each stock keeping unit (SKU) in a warehouse according to a variety of factors [Kofler et al., 2011].

Naturally, products are not always required at a uniform rate, and their demand often happens in waves. This is due to seasonality, product replacement, or marketing efforts as prensented in Carlo and Giraldo [2012]. The frequency of products reassignment varies from a company to another, depending on the type of industry. Thus, today's best product locations may be no longer optimal in a near future. In this case, it may be easy to compute the new desired situation, and identify which products should be moved to another location. However, one should still determine *how* to move products from the old to the new assignment. This is what we call the *warehouse reassignment problem*.

The storage strategy must be selected according to the picking method. Some policies do not consider product usage information, such as the random storage, while others, like class-based and full-turnover policies use the sales rate to determine the best location. For a review of classical assignment policies, the reader is directed to Gu et al. [2007] and De Koster et al. [2007]. It is important to understand these policies in order to assess the trade-off between a better assignment and the additional work generated by the movement of products. As reviewed in [Kofler et al., 2014], most studies in this area have been devoted to re-warehousing, which involves extensive rearrangements of all locations, such as the multi-period storage location assignment problem. In these tactical strategies [Rouwenhorst et al., 2000], it can be a better tradeoff to move just a subset of products, in a strategy known as healing [Kofler et al., 2011] in which we try to maximize the gain with a limited number of reassignments.

Chen et al. [2011] present a tabu search heuristic to relocate items in a warehouse by simultaneously deciding which ones are to be relocated and their destination in order to satisfy the required throughput during peak periods. They do not consider cycles and assume that the items to be relocated to destinations are decision variables. Carlo and Giraldo [2012] propose the rearrange-while-working strategy for unit-load storage. To do this, an AS/RS move the complete unit-load from a location to a workstation. When the picker has finished picking products, the remaining products of the unit-load is reassigned to a new empty location.

In this chapter we are concerned with determining the best way to reassign products to new locations in a classical warehouse. To the best of our knowledge, very few studies discuss the time workload part of the process. The reassignment theory was first proposed by Christofides and Colloff [1973]. They propose a two-stage algorithm to minimize the travel costs required to rearrange the products. The first stage identifies how each of the cycles can be repositioned. A cycle is composed of two or more products that exchange their positions. The second stage uses dynamic programming to determine the sequence in which the cycles are performed. Pazour and Carlo [2015] study the same problem (which they label the reshuffling concept) but relax the assumptions regarding having only cycles that must be executed separately. By this, they need to consider that open locations will change throughout the reasignment process. They propose a mathematical formulation for the reassignment problem in cycles.

There are a major differences between our problem settings and that of Christofides and Colloff [1973] and of Pazour and Carlo [2015]. The most important difference is that they only allow to drop a product at an empty shelf, whereas we allow a product switch incurring a time penalty.

Definition 1. *Products switch.* Let product A be an occupied location, and product B already on the vehicle to be dropped at the position currently occupied A. The product switch is defined as dropping B on the floor (near the new location), removing A and also setting it aside on the floor, picking up and placing B inside the now empty location, and picking up A to ove it towards its destination.

By relaxing this assumption we create a more complex problem. Nonetheless, we propose a simpler and more flexible solution which can be used with or without the assumption of dropping at empty locations.

The main contributions of this chapter are the development of exact methods for solving the warehouse reassignment problem that is still not widely studied in the literature. We present a new and original graph definition that allows great performance and flexibility. As will be demonstrated by our extensive computational experiments, our methods are very efficient and provide results close to optimality.

The remaining of this chapter is organized as follows. Section 4.2 presents the problem formulation and an illustrative example of the reassignment problem. Section 4.3 presents the mathematical models and a set of inequalities to strengthen the formulations in Section 4.4. Section 4.5 describes the computational details of our experiments, such as the software/hardware material, the instances generation, all the results from exact formulation methods and a simulation for a unit-load order picking system to showcase the benefits of the reassignment. Based on our observations, we also present an estimation of the working time of a reassignment corresponding to our instances with a current real-life technique. Finally, Section 4.6 concludes the chapter and presents some research perspectives.

4.2 Problem formulation

Following the formulation of Christofides and Colloff [1973] of the reassignment problem, let $A = [a_n]$, $n \in \{1, \ldots, N\}$ be the initial assignment of products to N locations, and let $B = [b_n]$, the desired final assignment. Thus, a_n and b_n are respectively the starting product and desired product at location n. We denote the I/O point as node 0.

Our problem applies to storage warehouse in which a product can be assigned to only one location at the time. If we have more than one pallet of a given product, we create as many dummy pallets required and equally divide the demand between them. Some locations are not occupied, and it is possible to move a product to an occupied location and operate a products switch. When this happens, it creates an additional handling time and forces to leave this location with the product that was previously there. All reassignment routes must begin and end at the I/O point (depot). The handling vehicle has a capacity of one pallet, and we assume that a product occupies an entire pallet and location. Then the warehouse reassignement problem can be formally define as follows. Given a unit-load warehouse with an initial assignment A, a desired assignment B and a maximum number of operators, the objective is to determine the set of routes followed be the pickers in such a way to minimize the total working time (traveling time, pick, drop and switch time).

Figure 4.1 shows an example of reassignment of products inside a warehouse. In the left part, we see the initial assignment A. The locations are colored according to the picking frequency of their products. From white, the less frequent, to black, the more frequently picked. Suppose that according to the fluctuation of demand and products availability, the new best assignment should be setup B, on the right side. We have to compute the material handling effort required to move from setup A to B.

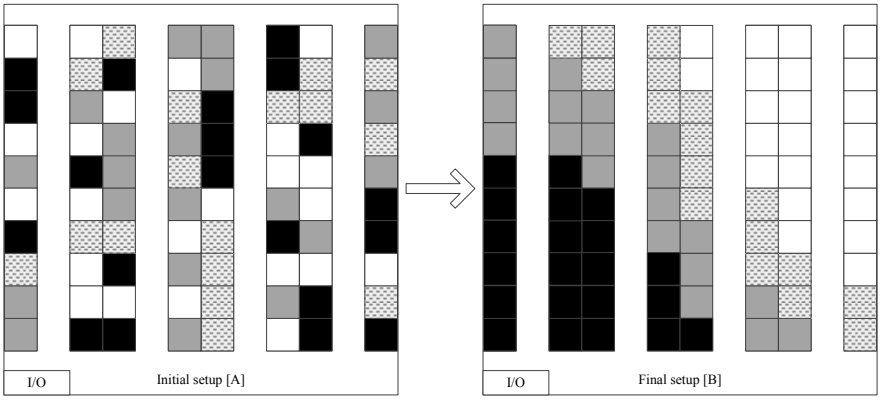


Figure 4.1 – Reorganization of products inside the picking area, colored by picking frequency

We consider that we always pick a product from its previous location and drop it directly to its desired location. In other words, there is not an intermediate movement that temporarily

places a product in a location that is not its final destination. The main difference with the problem solved by Christofides and Colloff [1973] is the fact that we accept to move a product to an occupied location. Obviously, this is not always the best alternative because the operator will need to switch the two products, which is costly in terms of handling time. To model this tradeoff, we add a penalty α to a drop at an occupied location that corresponds to dropping the new product on the floor (near the new location), removing the old one, placing the new product inside the location, and retake the old one.

We now present the following reduced example of a picking zone with only eight locations in order to understand all involved product movements. Each picking location is indexed by a number. The product index inside the location is indicated by a letter. The initial assignment is $A = [a, b, c, \emptyset, d, e, f, g]$ where $\emptyset$ represents an empty location. The desired final assignment is $B = [e, b, d, a, f, c, g, \emptyset]$. Note that only product b in location 2 stays in the same position. In Table 4.1, we show a quick route construction (not necessarily optimal) to move from an initial assignment A to a final assignment B.

The left part of the table shows step by step the six modifications done to the assignment until we reach the final one. In the right part, we show the evolution of the route with the location numbers. A single arrow ($\rightarrow$) means that we move empty between two locations. A double arrow ($\Rightarrow$) means that we move with a product. A left-right arrow ($\Leftrightarrow$) means that a product is dropped in an occupied location and we do a product switch. The first line of Table 4.1 shows the initial setup A, and each line corresponds to an intermediate assignment. The last line designates the desired assignment B. At each step, the moved product is in a gray cell.

Steps	Locations								Route
	1	2	3	4	5	6	7	8	
A	a	b	c	$\emptyset$	d	e	f	g	
S1	$\emptyset$	b	c	a	d	e	f	g	I/O $\rightarrow 1 \Rightarrow 4$
S2	e	b	c	a	d	$\emptyset$	f	g	I/O $\rightarrow 1 \Rightarrow 4 \rightarrow 6 \Rightarrow 1$
S3	e	b	$\emptyset$	a	d	c	f	g	I/O $\rightarrow 1 \Rightarrow 4 \rightarrow 6 \Rightarrow 1 \rightarrow 3 \Rightarrow 6$
S4	e	b	d	a	$\emptyset$	c	f	g	I/O $\rightarrow 1 \Rightarrow 4 \rightarrow 6 \Rightarrow 1 \rightarrow 3 \Rightarrow 6 \rightarrow 5 \Rightarrow 3$
S5	e	b	d	a	$\emptyset$	c	g	$\emptyset$	I/O $\rightarrow 1 \Rightarrow 4 \rightarrow 6 \Rightarrow 1 \rightarrow 3 \Rightarrow 6 \rightarrow 5 \Rightarrow 3 \rightarrow 8 \Leftrightarrow 7$
S6	e	b	d	a	f	c	g	$\emptyset$	I/O $\rightarrow 1 \Rightarrow 4 \rightarrow 6 \Rightarrow 1 \rightarrow 3 \Rightarrow 6 \rightarrow 5 \Rightarrow 3 \rightarrow 8 \Leftrightarrow 7 \Rightarrow 5 \rightarrow$ I/O
B	e	b	d	a	f	c	g	$\emptyset$	

Table 4.1 – Example of reassignment route construction

In the step S1, we move from the I/O point to location 1, picking product a and moving it to the empty location 4, letting location 1 temporarily empty ($\rightarrow 1 \Rightarrow 4$).

Since location 4 was not occupied before the move, we leave this location empty and we choose to move towards location 6. In the step S2, we pick product e and move it to its final location 1 ($\rightarrow 6 \Rightarrow 1$).

In step S3, we restart from location 1 and travel empty to location 3, picking c towards location 6, letting location 3 empty and filling location 6 ($\rightarrow 3 \Rightarrow 6$).

In step S4, we move empty from location 6 and pick product d at location 5, now temporarily empty, and bring it to the empty location 3 ($\rightarrow 5 \Rightarrow 3$).

In step S5 we go towards location 8, picking g and dropping it at location 7 which is occupied by product f. We hence have to take product f from the location, drop it on the floor, pickup product g that was already on the floor, drop it at location 8 and finally pick up product f again. We assume the penalty α associated with this move ($\rightarrow 8 \Leftrightarrow 7$).

In step S6, we move f to its final location 5 that we previously let empty and finish the route at the I/O ($\Rightarrow 5 \rightarrow$ I/O). We have achieved the final positioning B.

4.3 Mathematical model

This section presents different mathematical formulations for the warehouse reassignment problem.

Let $\mathcal{P}_i$ be a unit-load pick to perform at location $i \in N \backslash \{a_i = \emptyset\}$. We define the set of all picks $\mathcal{P} = \cup_{i \in N \backslash \{a_i = \emptyset\}} \mathcal{P}_i$. Let $\mathcal{D}_i$ be the drop to perform at location $i \in N \backslash \{b_i = \emptyset\}$. We also define the set of all drops $\mathcal{D} = \cup_{i \in N \backslash \{b_i = \emptyset\}} \mathcal{D}_i$. We define $\mathcal{R}_i$ as the destination (drop) associated with pick $\mathcal{P}_i$. Table 4.2 presents an example of components of set $\mathcal{R} = \cup_{i \in N} \mathcal{R}_i$. In this table, the product from location 1 has to be moved to location 2. Thus, location 2 corresponds to $\mathcal{R}_1$. $\mathcal{R}_2$ corresponds to location 1 as it is the destination of pick on location 2, thus $\mathcal{P}_2$.

Table 4.2 – Reassignment requests

N	1	2	3	4	5
1		$\mathcal{R}_1$			
2	$\mathcal{R}_2$				
3				$\mathcal{R}_3$	
4					$\mathcal{R}_4$
5			$\mathcal{R}_5$		

In order to formulate the problem, let u_i be the time at which the vehicle leaves location i. Other variables are model-specific and are presented with ech of the following three models. Let the time limit of a route be $\mathcal{L}$.

4.3.1 Three-nodes formulation (M1)

In this section we develop a graph definition especially designed for this problem. It will allow us to track information on the status of a location over time. We consider three types of actions that can be performed for each location, each represented by one node. The first two are related to dropping a product at the location or picking the product, represented by the sets $\mathcal{D}$ and $\mathcal{P}$. The third type of action is related to what happen after a drop in a empty

location. This means that we are leaving the location empty, without any product on the vehicle and without doing a product switch. For representing this action at each location, let $\mathcal{E}_i$ represent an empty node at location $i \in N \backslash \{b_i = \emptyset\}$. We define the set of all possible empty locations $\mathcal{E} = \cup_{i \in N \backslash \{b_i = \emptyset\}} \mathcal{E}_i$.

Definition 2. *Empty node.* For a location $i \in N \backslash \{b_i = \emptyset\}$, let an empty node $\mathcal{E}_i$ be the immediate successor of the drop node $\mathcal{D}_i$ if $\mathcal{P}_i = \emptyset$ or if $u_{\mathcal{P}_i} < u_{\mathcal{D}_i}$, meaning that the picker has previously placed a product at location i and left it without a product.

These three types of nodes per location are illustrated in Figure 4.2. This allows us to easily determine if a picker reachs a location with or without a product and if he leaves it with or without a product.

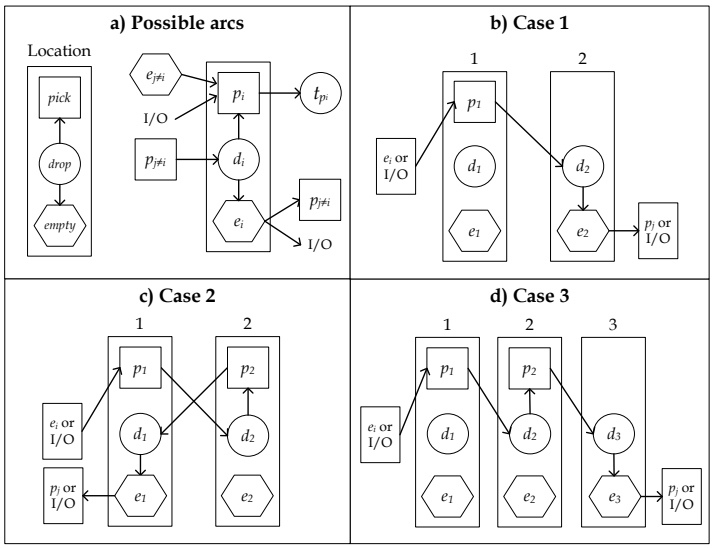


Figure 4.2 – The three possible cases of sequences of nodes

Figure 4.2 presents three possible sequences of nodes. A sequence starts and finishes without product and continues as long as the picker moves with a product.

In Figure 4.2a), we reach pick $\mathcal{P}_1$ in location 1, completing the reassignment request $\mathcal{R}_1$ by dropping it at its destination at location 2, corresponding to node $\mathcal{D}_2$. Since the destination was initially empty ($a_2 = \emptyset$), we just move to the empty node $\mathcal{E}_2$, completing the sequence for only one request.

In Figure 4.2b), the sequence starts and ends at the same location, creating a cycle. From pick $\mathcal{P}_1$, we go towards location 2 and node $\mathcal{D}_2$. Since there is already a product at this location, we have to go directly to $\mathcal{P}_2$. The destination of $\mathcal{P}_2$ is location 1, where we made $\mathcal{P}_1$ at the first step. Since we have emptied location 1, the path continues to the empty node $\mathcal{E}_1$, exiting location 1 without a product on the vehicle and ending the sequence. Two requests have been

satisfied in this example.

In Figure 4.2c), we have an extension of *case 1*. This is a cascade of requests within occupied locations. We start with $\mathcal{P}_1$. Its destination is drop $\mathcal{D}_2$ at the already occupied location 2. We hence need to pick $\mathcal{P}_2$ and so on, until we reach the drop node $\mathcal{D}_3$ in the empty location 3 that will end the sequence. This case shows how a starting pick node can generate a potentially long sequence of multiple requests.

Finally, let $G = (V, E)$ be the full graph, where $V = \mathcal{D} \cup \mathcal{E} \cup \mathcal{P} \cup 0$ is the set of all nodes and E is the set of all arcs x_{ij}, where $i \neq j$, developed as follows:

$$x_{ij} \in E \ if \begin{cases} i = 0 \text{ and } j \in \mathcal{P} & \text{(4.1a)} \\ i = \mathcal{E}_n \text{ and } j \in \mathcal{P}_m \cup 0 \quad \forall\ n, m \in N : n \neq m & \text{(4.1b)} \\ i = \mathcal{D}_n \text{ and } j = \mathcal{P}_n \quad \forall\ n \in N & \text{(4.1c)} \\ i = \mathcal{D}_n \text{ and } j = \mathcal{E}_n \quad \forall\ n \in N. & \text{(4.1d)} \end{cases}$$

In equation (4.1a) we have all arcs between the I/O point (node 0) and all picks. In (4.1b), we have arcs from empty nodes in $\mathcal{E}$ to pick nodes of different locations or returning to the I/O point. Equation (4.1c) sets the arc between the drop node and the pick node of the same location n (if a pick exists in this location). In the same way, equation (4.1d) sets the arc between the drop node and the empty node of the same location n.

The cost (time) to move from node i to another node j is c_{ij}, according to their respective location and such that $(i, j) \in E$. We consider a constant movement speed. When arc (i, j) corresponds to a product switch (arcs from equation (4.1c)), a penalty α is added to c_{ij}. For node $j \in \mathcal{P} \cup \mathcal{D}$ we define a service time s_j corresponding to the time to drop or pick the product. For the sake on simplicity, we also add s_j directly in c_{ij}.

We define the integer decision variable k as the number of pickers used in the solution. We define w_i as a continuous variable indicating the idle time at location i. Let $\mathcal{V}' = \mathcal{V} \backslash \{0\}$. Table 4.3 presents a summary of the parameters, sets and variables used in our formulation.

Table 4.3 – Summary of parameters, sets and variables

N	set of locations
c_{ij}	travel time for arc $(i,j) \in E$
$\mathcal{L}$	time limit of a route
$\mathcal{P}_n$	pick node of location $n \in N$
$\mathcal{D}_n$	drop node of location $n \in N$
$\mathcal{E}_n$	empty node of location $n \in N$
$\mathcal{R}_n$	destination of $\mathcal{P}_n$, $n \in N$
V	set of all nodes, $V = \mathcal{D} \cup \mathcal{E} \cup \mathcal{P} \cup 0$
E	set of arcs
x_{ij}	binary variable equal to 1 if the arc (i,j) is selected, 0 otherwise
u_i	departure time at node i, $i \in V^{'}$
w_i	idle time at node i, $i \in V^{'}$
k	number of routes in the solution

The mathematical formulation is the following:

$$Min\ Z = \sum_{(i,j)\in E} c_{ij}\ x_{ij} + \sum_{i\in V'} w_i \tag{4.2}$$

subject to:

$$\sum_{i\in\mathcal{E}} x_{i0} = k, \tag{4.3}$$

$$\sum_{j\in\mathcal{P}} x_{0j} = k, \tag{4.4}$$

$$\sum_{i\in V\setminus\{\mathcal{P}\}} x_{ij} = 1 \qquad \forall j \in \mathcal{P}, \tag{4.5}$$

$$\sum_{\substack{i\in V \\ |(i,j)\in E}} x_{ij} = \sum_{\substack{k\in V \\ |(j,k)\in E}} x_{jk} \qquad \forall j \in V^{'}, \tag{4.6}$$

$$u_i - u_j + \mathcal{L}x_{ij} \leq \mathcal{L} - c_{ij} \qquad \forall (i,j) \in E, \tag{4.7}$$

$$u_{\mathcal{P}_n} \leq u_{\mathcal{D}_n} + (1 - x_{\mathcal{D}_n,\mathcal{E}_n})\mathcal{L} \qquad \forall n \in N : \mathcal{P}_n \notin \emptyset, \tag{4.8}$$

$$u_j \leq u_i + w_j + c_{ij} + (1 - x_{ij})\mathcal{L} \qquad \forall (i,j) \in E, \tag{4.9}$$

$$x_{\mathcal{P}_n,\mathcal{R}_n} = 1 \qquad \forall n \in N, \tag{4.10}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \tag{4.11}$$

$$0 \leq u_i \leq \mathcal{L} \qquad \forall i \in V, \tag{4.12}$$

$$0 \leq w_i \leq \mathcal{L} \qquad \forall i \in V, \tag{4.13}$$

$$k \in \mathbb{N}. \tag{4.14}$$

The objective (4.2) minimizes the total workload corresponding to the sum of the traveling time (including penalties), the service time, and the idle time. Constraints (4.3) and (4.4) fix the number of routes that respectively exit and enter the I/O point. Constraints (4.5) ensure that all pick nodes will be visited and in the same way, performing all reassignment requests. Constraints (4.6) ensure the flow equilibrium at each node. Constraints (4.7) allow the chronometer increment of variable u_i considering the travel distance between i and j and the service time when applicable. This also removes all the possibilities of sub-tour within a solution. Constraints (4.8) ensure the pick node to be visited before the empty node of the same location. This constraint is valid if and only if the arc between the drop and empty of the same location $(\mathcal{D}_n, \mathcal{E}_n)$ is used. Constraints (4.9) are used to fix the idle time variable w_j if an arc x_{ij} is used. Constraints (4.10) impose that the arc between a pick in $\mathcal{P}_n$ towards its destination $\mathcal{R}_n$ must be used since we have a unit-load system. Constraints (4.11) set the nature of variable x_{ij}. Constraints (4.12) and (4.13) bound the variable u_i and w_i, respectively, to a maximal value $\mathcal{L}$. Constraint (4.14) indicates that variable k is a positive integer.

4.3.2 Vehicle-indexed formulation (M2)

In this section we present a vehicle-indexed adaptation of the previous formulation. We use a set of $\mathcal{M}$ pickers and for each $k \in \mathcal{M}$ we set a starting node k^+ and an ending node k^-, all corresponding to the I/O point. We then define $\mathcal{M}^+$ and $\mathcal{M}^-$ as the set of I/O nodes. Finally, let $W = \mathcal{M}^+ \cup \mathcal{M}^-$. We hence define variables x_{ij}^k equal to one if vehicle $k \in \mathcal{M}$ travels between i and j, zero otherwise. Let redefine the set of nodes $V = \mathcal{D} \cup \mathcal{E} \cup \mathcal{P} \cup W$ and $V' = V \backslash \{W\}$. Since we can now create a variable u_{k^+} and u_{k^-} for all $k \in \mathcal{M}$, we do not need idle time variables w_i to compute the total workload. Restrictions (4.1a) and (4.1b) for three index variables x_{ij}^k on set E are updated as follows:

$$x_{ij}^k \in E \ if \begin{cases} i \in \mathcal{M}^+,\ j \in \mathcal{P} \cup \mathcal{M}^- & \forall\ k \in \mathcal{M} \quad (4.15a) \\ i = \mathcal{E}_n,\ j \in \mathcal{P}_m \cup \mathcal{M}^- & \forall\ n, m \in N : n \neq m, k \in \mathcal{M} \quad (4.15b) \end{cases}$$

The model is the following:

$$Min\ Z = \sum_{k \in \mathcal{M}} (u_{k^-} - u_{k^+}) \tag{4.16}$$

subject to:

$$\sum_{i \in \mathcal{E} \cup \{k^+\}} x_{ik^-}^k = 1, \qquad \forall k \in \mathcal{M}, \tag{4.17}$$

$$\sum_{j \in \mathcal{P} \cup \{k^-\}} x_{k^+j}^k = 1, \qquad \forall k \in \mathcal{M}, \tag{4.18}$$

$$\sum_{k\in\mathcal{M}}\sum_{i\in V'\cup\mathcal{M}^+\setminus\{\mathcal{P}\}} x_{ij}^k = 1 \qquad \forall j\in\mathcal{P}, \tag{4.19}$$

$$\sum_{\substack{i\in V'\cup\mathcal{M}^+\\ |(i,j)\in E}} x_{ij}^k = \sum_{\substack{l\in V'\cup\mathcal{M}^-\\ |(j,l)\in E}} x_{jl}^k \qquad \forall j\in V', k\in\mathcal{M}, \tag{4.20}$$

$$u_i - u_j + \mathcal{L}x_{ij}^k \leq \mathcal{L} - c_{ij} \qquad \forall (i,j,k)\in E, \tag{4.21}$$

$$u_{\mathcal{P}_n} \leq u_{\mathcal{D}_n} + (1 - \sum_{k\in\mathcal{M}} x_{\mathcal{D}_n,\mathcal{E}_n}^k)\mathcal{L} \qquad \forall n\in N : \mathcal{P}_n \notin \emptyset, \tag{4.22}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j,k)\in E, \tag{4.23}$$

$$0 \leq u_i \leq \mathcal{L} \qquad \forall i\in V. \tag{4.24}$$

The objective (4.16) minimizes the total workload by taking the difference between the ending and starting time for all vehicles. Constraints (4.17) and (4.18) make sure that each vehicle starts and ends at the I/O point. Constraints (4.19) ensure that each pick node will be visited only once. Constraints (4.20) ensure the flow equilibrium at a node. Constraints (4.21) allow the chronometer increment of variable u_i. Constraints (4.22) ensure the pick to be visited before the empty node of the same location when applicable. Constraints (4.23) and (4.24) define the nature of variables x_{ij}^k and u_i respectively.

4.3.3 Pickup and delivery formulation (M3)

This section presents how a general pickup delivery formulation [Savelsbergh and Sol, 1995] can be adapted to solve the reassignment problem. We use four types of variables. Variables z_i^k for each $i\in\mathcal{P}, k\in\mathcal{M}$ equal to 1 if pick i is assigned to vehicle k, 0 otherwise. Variables x_{ij}^k such that $(i,j)\in(V'\times V')\cup\{(k^+,j|j\in\mathcal{P})\}\cup\{(j,k^-)|j\in\mathcal{D}\}, k\in\mathcal{M}$ equal to 1 if vehicle k travels from location i to location j, 0 otherwise. We still use variables u_i as the departure time from node $i\in V\cup W$. Let y_i be the load of vehicle at node $i\in V\cup W$. The mathematical formulation is as follows:

$$Min\ Z = \sum_{k\in\mathcal{M}} (u_{k^-} - u_{k^+}) \tag{4.25}$$

subject to:

$$\sum_{k\in\mathcal{M}} z_i^k = 1 \qquad \forall i\in\mathcal{R}, \tag{4.26}$$

$$\sum_{j\in V} x_{lj}^k = \sum_{j\in V} x_{jl}^k = z_i^k \qquad \forall n\in N, l\in\mathcal{P}_n\cup\mathcal{R}_n, k\in\mathcal{M}, \tag{4.27}$$

$$\sum_{j\in V'\cup\{k^-\}} x_{k^+j}^k = 1 \qquad \forall k\in\mathcal{M}, \tag{4.28}$$

$$\sum_{j \in V' \cup \{k^+\}} x^k_{ik^-} = 1 \qquad \forall k \in \mathcal{M}, \tag{4.29}$$

$$u_p \leq u_d \qquad \forall n \in \mathcal{N}, p \in \mathcal{P}_n, d \in \mathcal{R}_n, \tag{4.30}$$

$$u_i - u_j + \mathcal{L} x^k_{ij} \leq \mathcal{L} - c_{ij} \qquad \forall (i,j) \in E, k \in \mathcal{M}, \tag{4.31}$$

$$y_{k^+} = 0 \qquad \forall k \in \mathcal{M}, \tag{4.32}$$

$$y_i \leq \sum_{k \in \mathcal{M}} z^k_i \qquad \forall i \in \mathcal{P}, \tag{4.33}$$

$$y_i - y_j + x^k_{ij} \leq 0 \qquad \forall i,j \in V, k \in \mathcal{M}, \tag{4.34}$$

$$u_{\mathcal{D}_n} - u_{\mathcal{P}_n} - (1 - \sum_{k \in \mathcal{M}} x^k_{\mathcal{D}_n \mathcal{P}_n}) \mathcal{L} \leq c_{\mathcal{D}_n \mathcal{P}_n} \qquad \forall n \in N : p_n \neq \emptyset, \tag{4.35}$$

$$u_{\mathcal{D}_n} - u_{\mathcal{P}_n} + \sum_{k \in \mathcal{M}} x^k_{\mathcal{D}_n \mathcal{P}_n} \mathcal{L} \geq c_{\mathcal{D}_n \mathcal{P}_n} \qquad \forall n \in N : p_n \neq \emptyset, \tag{4.36}$$

$$x^k_{ij} \in \{0,1\} \qquad \forall i,j \in V \cup W, k \in \mathcal{M}, \tag{4.37}$$

$$z^k_i \in \{0,1\} \qquad \forall i \in \mathcal{P}, k \in \mathcal{M}, \tag{4.38}$$

$$u_i \geq 0 \qquad \forall i \in V \cup W, \tag{4.39}$$

$$y_i \geq 0 \qquad \forall i \in V \cup W. \tag{4.40}$$

The objective function (4.25) minimizes the total workload for all vehicles. Constraints (4.26) ensure that all pick requests will be served only once. With constraints (4.27), a vehicle enters or leaves a location l if it is a pick or a drop of a transportation request assigned to that vehicle. By constraints (4.28) and (4.29), we make sure that each vehicle starts and ends at the I/O point. Constraints (4.30) ensure that the pick is made before the drop of the same request. Constraints (4.31) ensure the correct increment of the departure time variables when an arc is used. Constraints (4.32) and (4.33) impose the initial and maximum load of the vehicle respectively. Constraints (4.34) make the correct increment of the load when applicable. Together, constraints (4.35) and (4.36) ensure that product switch at the same location will be done if a drop is made at a still occupied location. Constraints (4.37) to (4.40) define the nature of all involved variables.

4.4 Models lifting

In this section, we present how it is possible to strengthen the bound of timing variables for all models *M1*, *M2* and *M3*. We also show how it is possible to remove the symmetry of the vehicle-indexed formulations, *M2* and *M3*.

Between the starting point and the arrival point, a minimum path corresponds to a single reassignment request. We can tight the bound of u_i variables for all pick $\mathcal{P}$. Inequalities

(4.41) and (4.42) are valid for all locations $n \in N$ such that $p = \mathcal{P}_n$, $d = \mathcal{R}_n$.

$$c_{0p} \leq u_p \leq \mathcal{L} - c_{pd} - c_{d0} \tag{4.41}$$

$$c_{0p} + c_{pd} \leq u_d \leq \mathcal{L} - c_{d0}. \tag{4.42}$$

The Miller-Tucker-Zemlin constraints (4.7) can be lifted as presented by Kara et al. [2004] by reducing the maximal value of $\mathcal{L}$. This can be done as follows, by considering the minimal travel time to the node i:

$$u_i - u_j + (\mathcal{L} - min_k\{c_{ki}\})x_{ij} \leq \mathcal{L} - min_k\{c_{ki}\} - c_{ij} \quad \forall (i,j) \in E. \tag{4.43}$$

It is also possible to determine an initial valid lower bound considering that each reassignment request must be made. The lower bound is the following:

$$Z \geq + \sum_{n \in N} c_{\mathcal{P}_n \mathcal{D}_n} + a \left(\min_{p \in \mathcal{P}} \{c_{0p}\} + \min_{d \in \mathcal{D} \cup \mathcal{E}} \{c_{d0}\} \right) \tag{4.44}$$

$$a = \left\lceil \frac{\sum_{n \in N} c_{\mathcal{P}_n \mathcal{D}_n}}{\mathcal{L}} \right\rceil. \tag{4.45}$$

Equation (4.45) states the minimum number of vehicles needed to cover the total travel times between a pick and its destination, including all the service time. Knowing this number, we know that solution will at least perform this amount of work to cover the minimal distance between the I/O and first (and last) nodes. Inequalities (4.41) – (4.44) are valid for all three formulations.

An important weakness of a homogeneous vehicle-indexed formulation (*M2* and *M3*) is the presence of solutions symmetry. We tighten these formulations by imposing the following symmetry breaking constraints:

$$\sum_{j \in P \cup \{k^-\}} x_{0j}^k \leq \sum_{j \in P \cup \{k^-\}} x_{0j}^{k-1} \quad \forall k \in \mathcal{M} \backslash \{1\} \tag{4.46}$$

$$\sum_{\substack{l \in V \\ |(l,i) \in E}} x_{li}^k \leq \sum_{\substack{c \in V \\ |(c,j) \in E}} \sum_{j<i} x_{cj}^{k-1} \quad \forall i \in V', k \in \mathcal{M} \backslash \{1\}. \tag{4.47}$$

Constraints (4.46) ensure that vehicle k cannot leave the depot if the vehicle $k-1$ is not used. This symmetry breaking rule is then extended to the locations by constraints (4.47) which states that if a request i is assigned to vehicle k, then vehicle $k-1$ must perform a request with an index smaller than i [Coelho and Laporte, 2013].

4.5 Computational experiments

In this section, we provide details on the implementation, benchmark instances, and describe the results of extensive computational experiments. The description of the benchmark instances is presented in Section 4.5.1. In Section 4.5.2 we describe how we have evaluated and estimated the current solution of an industrial partner. This is followed by the results of our computational experiments in Section 4.5.3. Finally, Section 4.5.4 presents a return over the investment analysis, in terms of working time, of the reassignment process on a unit-load picking warehouse.

We use IBM CPLEX Concert Technology 12.6 as the branch-and-bound solver. All computations were executed on machines equipped with Intel Westmere EP X5650 six-core processors running at 2.667 GHz, and with up to 16 GB of RAM running the Scientific Linux 6.3. All algorithms were given a time limit of 3600 seconds.

4.5.1 Instances generation

An instance is a set of positions inside the warehouse, represented by an aisle number (a) and a section number (s). There is a number of empty locations (e) randomly positioned within the warehouse. A unique product is located at each non-empty location, representing the initial setup of the warehouse. Finally, each product is assigned to a new location. The number of aisles is $a = \{1, 2, 3\}$, the number of sections is $s = \{2, 3, 4, 5\}$ and the number of empty locations is $e = \{\lceil 0.1(2as) \rceil, \lceil 0.2(2as) \rceil, \lceil 0.3(2as) \rceil\}$. That makes a total of 30 different instances, corresponding to one instance per configuration. Note that if two or more configurations with one aisle and the same number of sections lead to the same number of empty locations (e), we only create one instance. For example, one aisle and two section, the rounded up number of empty locations is always one for all proportion.

We also vary the experiments via a time limit ($\mathcal{L}$) of a reassignment route. This limit, in seconds, will be in $\mathcal{L} = \{\infty, 1000\}$. We set the time penalty (α) to 30 seconds. The lift trucks have a constant speed of 1 m/s. We assume a service time (s_i) of 10 seconds for all pick and drop nodes.

4.5.2 Real-case solution estimation

In order to validate the potential gain of the reassignment technique, it is appropriate to compare it against a real reassignment method. We will compare our method with that observed from a partner working in the industry of large volume food distribution. They have recently relocated all the products in their picking area. To do this, they removed all involved products from each aisle and put them in the consolidation zone, between the aisles and the docks. Afterwards, they positioned each product in its new location from this buffer storage

space using one lift truck operator per aisle. We can easily compute the cumulative time of this operation using or travel time matrix c_{ij}, including the service time.

It is assumed that products, once removed from their original location, are positioned just in front of their respective initial aisle. We will neglect the movements and distances around the buffer zone. It is assumed that all products must be removed before starting the reassignment. To calculate the total work time, we compute four movements per product. The first one is between the buffer zone of the front of the aisle and the product. The second is the same distance, but in the opposite direction. The third (and fourth) between the buffer zone of the initial aisle and the new location (and way back in opposite direction). This is done for each request to obtain an estimate of the total time to relocate all products. We will call this method the *Case Study Heuristic* (CSH).

4.5.3 Results

This section presents the computational experiments of all three mathematical models over the benchmark instances and a comparison in terms of performance and characteristics. Table 4.4 presents these first results. The first three columns indicate the number of aisles, sections and empty locations, respectively. The fourth column shows the number of reassignment requests. The CSH column presents the results of the case study heuristic. As computing times are negligible, they are not reported. For each mathematical formulation (M1, M2 and M3), the table reports the upper bound (UB), the lower bound (LB), the optimality gap (Gap (%)) and the CPU time in seconds (Time (s)). All models are reported with all their respective valid inequalities

We see that optimal solutions have been found for all instances with only one aisle ($a = 1$) for all three formulations. For instances with two aisles, formulation M2 and M3 begin to have difficulties to close the gap within the allotted time. Formulation M1 performs a lot better and finds optimal solutions for instances with up to three aisles. The total average optimality gap for M1 is only 2.9%. The gaps are a lot larger for M2 and M3 with 18.2% and 18.1% respectively. The best overall upper bound comes from M1 with 753.7. It corresponds to an improvement of 56% from the solutions of the case study heuristic. Formulation M1 finds the best lower bound for all instances. This formulation obtained 29 out of 30 best upper bounds. We see that formulations M2 and M3 have almost the same performance in terms of upper and lower bounds, gap and CPU times.

Table 4.5 presents the upper bound and lower bound of all three models without any timing valid inequalities (4.41) and (4.42), initial lower bound (4.44) and symmetry breaking inequalities (4.46) and (4.47). We see that M1 still found 20 out of 30 optimal solutions and gives very similar bounds. For both models M2 and M3, we see that after 9 requests, they are unable to find any valid lower bounds. That confirms the importance of inequalities proposed

Table 4.4 – Heuristic and models performances comparison

Instances				CSH	M1				M2				M3			
a	s	e	$\|\mathcal{R}\|$		UB	LB	Gap (%)	Time (s)	UB	LB	Gap (%)	Time (s)	UB	LB	Gap (%)	Time (s)
1	2	1	3	290	160	160	0.0	0	160	160	0.0	0	160	160	0.0	0
	3	1	5	515	280	280	0.0	0	280	280	0.0	0	280	280	0.0	0
	4	1	6	840	410	410	0.0	1	410	410	0.0	2	410	410	0.0	1
		2	6	740	300	300	0.0	0	300	300	0.0	0	300	300	0.0	0
	5	1	7	1235	500	500	0.0	29	500	500	0.0	105	500	500	0.0	94
		2	7	1070	400	400	0.0	1	400	400	0.0	11	400	400	0.0	15
		3	7	945	350	350	0.0	0	350	350	0.0	2	350	350	0.0	2
2	2	1	8	680	330	330	0.0	0	330	330	0.0	33	330	330	0.0	19
		2	9	640	330	330	0.0	0	330	330	0.0	3	330	330	0.0	3
	3	1	9	1265	610	610	0.0	1453	630	430	31.7	3600	640	430	32.8	3600
		2	9	1100	510	510	0.0	17	520	365	29.8	3600	520	365	29.8	3600
		3	10	1085	480	480	0.0	0	480	480	0.0	3297	480	480	0.0	3600
	4	1	10	2050	900	790	12.2	3600	910	655	28.0	3600	910	655	28.0	3600
		3	11	1700	680	680	0.0	1651	710	530	25.4	3600	710	530	25.4	3600
		4	11	1600	640	640	0.0	1	670	540	19.4	3600	670	540	19.4	3600
	5	2	12	2550	950	872	8.2	3600	1070	720	32.7	3600	1020	720	29.4	3600
		4	13	2290	930	851	8.5	3600	940	705	25.0	3600	980	705	28.1	3600
		6	13	1960	760	760	0.0	80	780	595	23.7	3600	760	595	21.7	3600
3	2	1	14	1240	650	650	0.0	875	670	475	29.1	3600	670	475	29.1	3600
		2	15	1150	610	610	0.0	13	640	475	25.8	3600	620	475	23.4	3600
		3	15	1010	510	510	0.0	4	500	380	24.0	3600	510	380	25.5	3600
	3	1	16	2165	1060	921	13.1	3600	1140	785	31.1	3600	1120	785	29.9	3600
		3	17	1895	860	860	0.0	985	900	685	23.9	3600	910	685	24.7	3600
		5	17	1695	840	840	0.0	1997	860	630	26.7	3600	870	630	27.6	3600
	4	2	18	3180	1550	1339	13.6	3600	1600	1180	26.3	3600	1630	1180	27.6	3600
		4	20	2940	1380	1221	11.5	3600	1450	1060	26.9	3600	1480	1060	28.4	3600
		7	21	2280	1030	1030	0.0	496	1120	780	30.4	3600	1070	780	27.1	3600
	5	3	22	4175	1720	1477	14.1	3600	1900	1270	33.2	3600	1850	1270	31.4	3600
		6	24	3700	1470	1385	5.8	3600	1620	1180	27.2	3600	1600	1180	26.3	3600
		9	27	3245	1410	1393	1.2	3600	1470	1100	25.2	3600	1520	1100	27.6	3600
	Average			1707.7	**753.7**	**716.3**	**2.9**	**1333.5**	788.0	602.7	18.2	2515.2	786.7	602.7	18.1	2524.5

in Section 4.4.

The graph of Figure 4.3 presents the number of variables used for each formulation as a function of the number of reassignment requests given in the fourth column of Table 4.4. Instances with less than 7 requests has a restriction of only one vehicle. Since M2 and M3 are both vehicle indexed formulation, it is normal to see that the number of variables is very similar with M1 under 7 requests. For instances with 8 to 13 requests, two vehicles are allowed and three vehicles from 14 requests. The graph shows a rapid increase in the number of variables for M2 and M3.

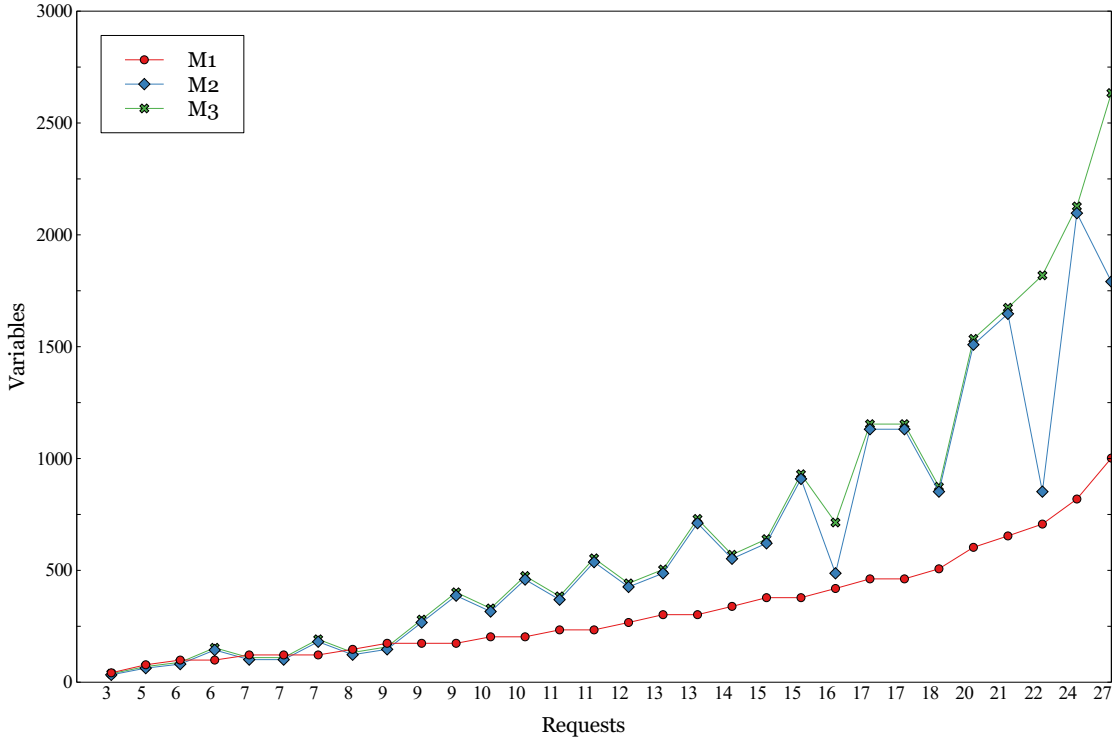


Figure 4.3 – Number of variables

Figure 4.4 presents the number of constraints for each formulation. For instances with 9 requests and more, the constraints number of M3 is rapidly increasing. For formulations with nodes per location (M1 and M2), the number of constraints increases less rapidly.

Figure 4.5 shows the time in milliseconds (ms) spent on average per branch-and-bound node for solving each relaxed problem. Again, from 9 requests the performance of formulation M1 is better than the other ones for almost all instances. This means that M1 is able to explore more nodes in the process and available time.

These results clearly demonstrate how the introduced formulation outperforms the other two, in particular the pickup and delivery formulation. With M1, we are able to solve to optimality instances with up to 17 reassignment requests in a reasonable amount of time. Formula-

Table 4.5 – Model performances without inequalities and initial lower bound

Instances				M1				M2				M3			
a	s	e	$\lvert\mathcal{R}\rvert$	UB	LB	Gap (%)	Time (s)	UB	LB	Gap (%)	Time (s)	UB	LB	Gap (%)	Time (s)
1	2	1	3	160	160	0.0	0	160	160	0.0	0	160	160	0.0	0
	3	1	5	280	280	0.0	0	280	280	0.0	0	280	280	0.0	0
	4	1	6	410	410	0.0	1	410	410	0.0	4	410	410	0.0	6
		2	6	300	300	0.0	0	300	300	0.0	1	300	300	0.0	1
	5	1	7	500	500	0.0	31	500	500	0.0	287	500	500	0.0	393
		2	7	400	400	0.0	1	400	400	0.0	23	400	400	0.0	31
		3	7	350	350	0.0	0	350	350	0.0	3	350	350	0.0	4
2	2	1	8	330	330	0.0	0	330	330	0.0	70	330	330	0.0	153
		2	9	330	330	0.0	0	330	330	0.0	8	330	330	0.0	14
	3	1	9	610	597	2.2	3600	620	-∞	-∞	3600	630	-∞	-∞	3600
		2	9	510	510	0.0	16	510	-∞	-∞	3600	510	-∞	-∞	3600
		3	10	480	480	0.0	0	480	-∞	-∞	3600	480	-∞	-∞	3600
	4	1	10	880	790	10.2	3600	930	-∞	-∞	3600	940	-∞	-∞	3600
		3	11	680	680	0.0	1754	720	-∞	-∞	3600	710	-∞	-∞	3600
		4	11	640	640	0.0	1	670	-∞	-∞	3600	670	-∞	-∞	3600
	5	2	12	950	875	7.9	3600	1040	-∞	-∞	3600				3600
		4	13	930	860	7.5	3600	970	-∞	-∞	3600	970	-∞	-∞	3600
		6	13	760	760	0.0	80	780	-∞	-∞	3600	770	-∞	-∞	3600
3	2	1	14	650	650	0.0	2238	690	-∞	-∞	3600	680	-∞	-∞	3600
		2	15	610	610	0.0	18	610	-∞	-∞	3600	620	-∞	-∞	3600
		3	15	510	510	0.0	5	500	-∞	-∞	3600	510	-∞	-∞	3600
	3	1	16	1070	921	13.9	3600	1110	-∞	-∞	3600	1150	-∞	-∞	3600
		3	17	860	860	0.0	1101	910	-∞	-∞	3600	950	-∞	-∞	3600
		5	17	840	840	0.0	2833	870	-∞	-∞	3600	860	-∞	-∞	3600
	4	2	18	1520	1335	12.2	3600	1620	-∞	-∞	3600				3600
		4	20	1400	1221	12.8	3600	1480	-∞	-∞	3600	1490	-∞	-∞	3600
		7	21	1030	1030	0.0	1364	1090	-∞	-∞	3600	1110	-∞	-∞	3600
	5	3	22	1700	1470	13.5	3600				3600				3600
		6	24	1500	1382	7.9	3600	1800	-∞	-∞	3600	1750	-∞	-∞	3600
		9	27	1410	1390	1.4	3600	1540	-∞	-∞	3600	1710	-∞	-∞	3600
Average				753.3	715.7	5.0	1514.8	758.6	-	-	2533.2	724.8	-	-	2540.1

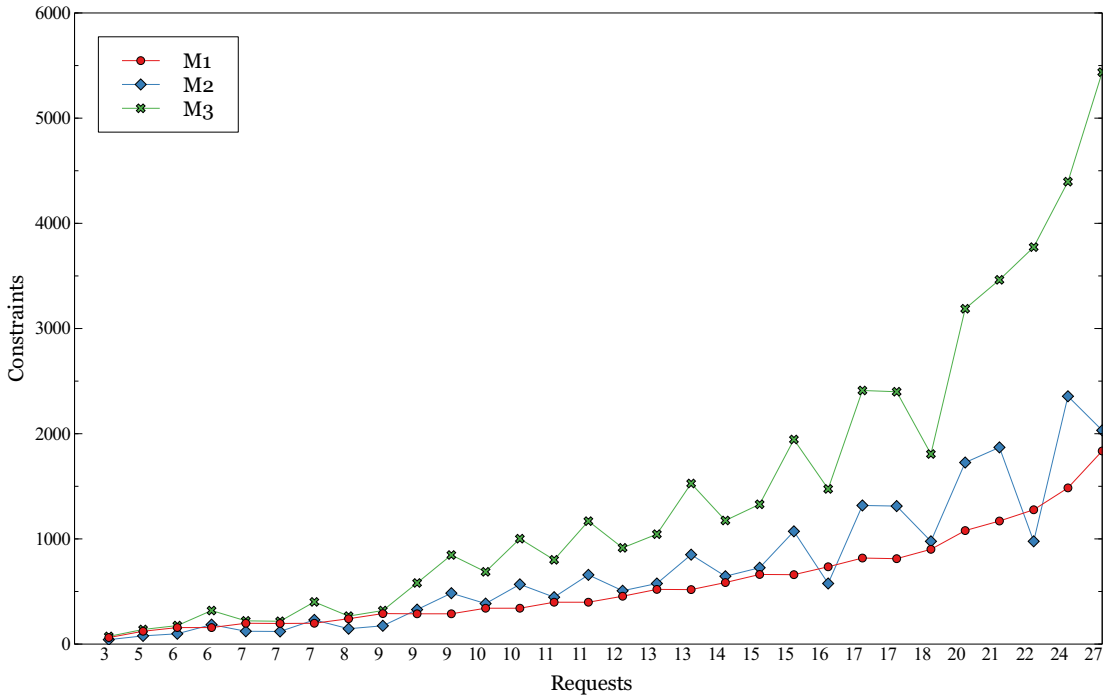


Figure 4.4 – Number of constraints

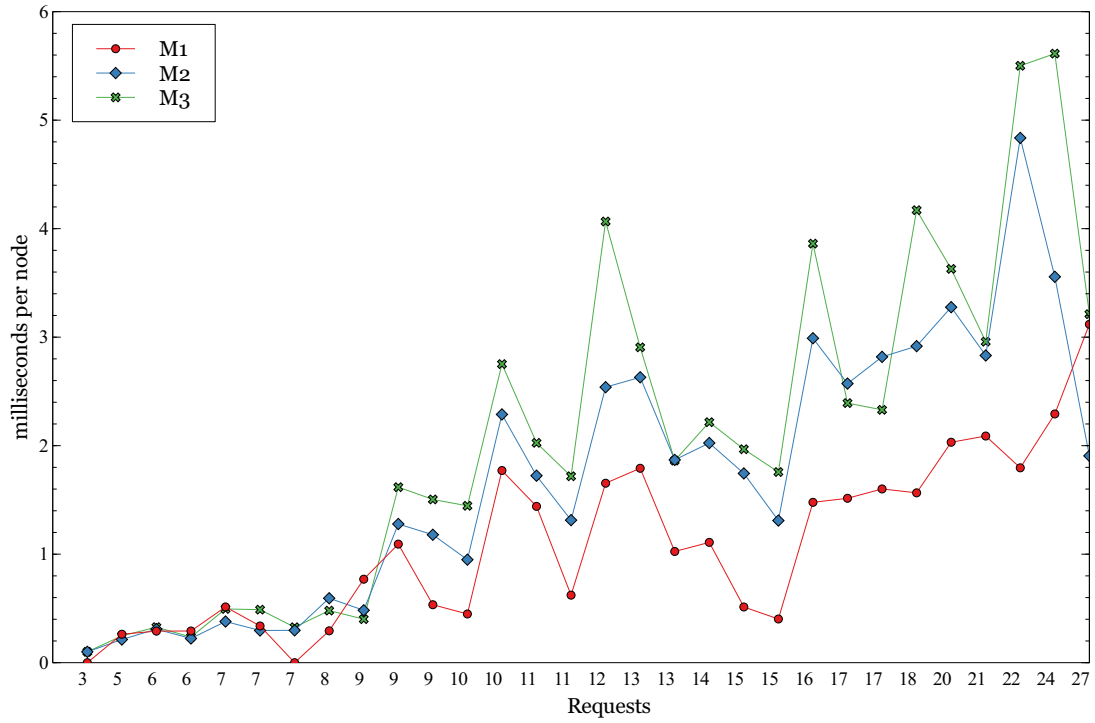


Figure 4.5 – Milliseconds passed per visited nodes

tions M2 and M3 have not been able to solve instances with more than 9 requests and their performance declines quickly after this threshold.

Recall that we allow two vehicles for instances with two aisles and three vehicles for instances with three aisles. For most instances, the solution tends to use all available vehicles. The main reason is because there will be fewer product switch involving a time penalty. For example, a second vehicle leaving the depot will directly pick a product at a location, enabling the first one to drop without a penalty. Coordinating several vehicles in the reassignment process thus presents a real advantage. We tested to reduce the time available for the vehicles for instances with two and more aisles. When a feasible solution is found, it leads to a similar distance to the solution without time capacity.

4.5.4 Simulation on a unit-load picking system

The reassignment process implies an important decision, since picking operations must be delayed (or strongly disturbed) while products are repositioned. For this reason, sometimes companies might hesitate to state the reassignment. However, one must consider that a bad assignment incurs higher picking cost/time. Thus, the reassignment should be seen as an investment whose value can be determined. To do this we simulate scenarios on the most basic picking system: a unit-load. We compute the total distance to pick all products from the pick list by making a round trip from the I/O point and the location of the product. The total distance before and after the reassignment can therefore be easily computed.

We generate picks list from 1 to 150 picks on the instance with 27 requests, such as the one on the last row of Table 4.4. Figure 4.6 presents the results of our simulation. It shows the total distance of the unit-load picking with and without the reassignment. Table 4.4 gives us the CSH total distance to reassign products and the distance from the best model (M1) that are not impacted by the long size of the picks list. It also shows the distance saved as the difference with the picking distance without and with the reassignment.

This allows us to delimit the gray area on the graph corresponding to the gain between our method and the CSH. Moreover, it shows that the distance saving is greater than the reassignment distance of our method at around 45 picks to do. In comparison, the saving distance because greater than the CSH distance after 110 picks. Since it is a very small warehouse example, this difference is important.

4.6 Conclusion

In this chapter we have suggested a new formulation for the warehouse reassignment problem. We have then been able to solve instances in which we have to relocate a large set of products within the picking zone. Our new directed graph definition and model minimize the workload

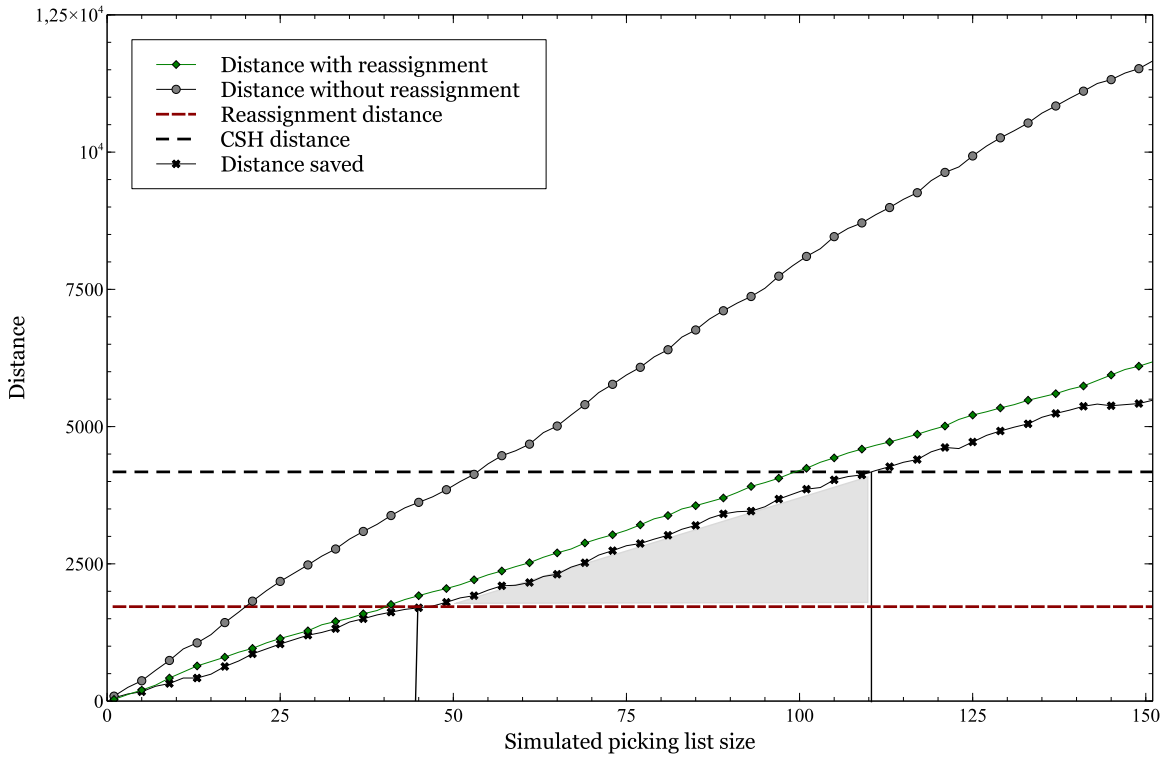


Figure 4.6 – Simulation of picking scenarios

of relocating all the products in their new position. We have seen that the model is very efficient and allows to solve instances of realistic size with handling movement penalties. We have generated a dataset of benchmark instances for the reassignment problem. As shown, companies may opt for simpler methods, but which dramatically requires more operation time and material handling. In comparison with a technique already used by an industrial partner, we can reduce on average by three times the workload of reassignment. We have been able to obtain a tight gap in most instances and for all given time capacity. Moreover, we have shown that on unit-load warehouse, the reassignment cost quickly pays off as the picking process becomes much more efficient. As future research and practice opportunities, we see that combining picking with reassignment operations can yield even higher savings.

Conclusion

Warehouse management has become an intensive field of research with the expansion of e-commerce. DCs have to continuously improve their operation to face a constant growth of demand. Several DCs activities can be optimized in order to improve the productivity which is often calculated in terms of order lead time. Many warehouse design decisions have a direct impact on order picking activities. We have shown that the order picking is influenced by, among others, the facility layout, the storage assignment policy, the routing strategy, etc. With today's computational capacity, it remains hard to integrate all these choices in the order picking optimization. This is why most of research continues to focus on one or two aspects of this large spectrum. Following the rich literature on material handling, we have introduced rich order picking problems and a new and efficient product reassignment formulation.

In Chapter 2, we have modeled an order picking problem with several layout and equipment constraints based on the warehouse of an industrial partner. The problem takes place in a narrow-aisle layout environment. We have shown the value of considering the high-level picking aspect in the resolution process. The narrow-aisle configuration of the warehouse allows us to decompose the picking problem per aisle. Even with this decomposition, the movements and capacity constraints make this problem hard to solve to optimality. We have proposed heuristic methods and exact formulations to solve large instances. An exact method based on an adapted formulation of the vehicle routing problem determines the items batching and routing. This is done because we can use the picking area configuration to improve the formulation of the model. With movement constraints, we strengthen the exact formulation into an acyclic one.

Chapter 3 presents another order picking problem motivated by a situation prevailing in the grocery industry. Several product families are present and the characteristics of each product have to be considered. We have included three main practical constraints from real-life observations. The first one is the weight constraint in which, after a certain threshold, it becomes impossible to add products with a weight over the acceptable limit. The second constraint is the fragility one that limits the weight that can be put over a product on the pallet to its self-capacity. The last one is the category constraint in which we separated products into two main families: food and non-food. We forbid placing a non-food product

over a food one. Each product characteristic considered in the problem adds an extra layer of difficulty. We have solved this problem as a rich routing problem and we have presented several classical order picking methods, a metaheuristic and two exact formulations. We have shown that most constraints generated by product characteristics can be dynamically added to models with a solid branch-and-cut scheme. As expected, our computational experiments showed that classical heuristics are not well adapted for this family of routing problem and can give very poor results. In return, a richer heuristic with local searches outperforms these classical ones. Our exact models are very good to yield near-optimal solutions in a reasonable time.

In Chapters 2 and 3 we have considered fixed product assignments within a DC. It has been proved that updating and improving product assignments offer significant benefits. Many reasons exist on why we should update the product locations: product lifetime, promotion campaigns, seasonality, etc. Changing product locations is a very labor intensive activity and can block the order picking operation for a long period of time. Chapter 4 presents an original formulation of the reassignment problem based on the possible operations to be performed at each location. The problem takes place over a planning horizon, represented by the time allotted to the operator to perform a reassignment route. Precedence constraints and the timing perspective make this problem very hard to solve. We have created a benchmark of reassignment instances that we solve with our methods. In addition, we have compared our new formulation against a modified one and a pickup and delivery vehicle routing formulation. Our computational analysis have demonstrated that our new formulation almost dominates the other ones in terms of results, and also in terms of the number of variables, constraints and time spent at each branch-and-bound node.

This thesis contributes to the literature in a number of ways. We have narrowed the gap between a part of the theoretical approaches existing in the literature about material handling and a more real-life based methodology which takes into account several realistic constraints. As an example, Chapter 3 has been published in a special issue in *warehouses design and management* which shows new trends in the DC management [De Koster et al., 2017]. Through out this thesis, we highlight the importance of practical constraints within the resolution methods and we show how to use them to strengthen formulations. These constraints come from the layout configuration (narrow-aisles and picker safety), products characteristics (weight, fragility and category) to the operational nature of the operator movements which inspired the new proposed reassignment formulation.

The ever changing world of distribution and new technologies offer new challenges. On the future, we believe that more research will be needed on practical and dynamic applications.

Bibliography

K. B. Ackerman. Practical Handbook of Warehousing. Springer Science & Business Media, New York, 2013.

A. Agarwal, R. Shankar, and M. K. Tiwari. Modeling the metrics of lean, agile and leagile supply chain: An anp-based approach. European Journal of Operational Research, 173(1): 211 – 225, 2006.

J. Ashayeri and L. F. Gelders. Warehouse design optimization. European Journal of Operational Research, 21(3):285–294, 1985.

P. Baker and M. Canessa. Warehouse design: a structured approach. European Journal of Operational Research, 193(2):425–436, 2009.

J. J. Bartholdi and S. T. Hackman. Warehouse & distribution science: release 0.92. Atlanta, GA, The Supply Chain and Logistics Institute, School of Industrial and Systems Engineering, Georgia Institute of Technology, 2011.

D. Battini, M. Calzavara, A. Persona, and F. Sgarbossa. Additional effort estimation due to ergonomic conditions in order picking systems. International Journal of Production Research, 55(10):2764–2774, 2017.

P. Bodnar and J. Lysgaard. A dynamic programming algorithm for the space allocation and aisle positioning problem. Journal of the Operational Research Society, 65(9):1315–1324, 2013.

M. Bortolini, M. Faccio, M. Gamberi, and R. Manzini. Diagonal cross-aisles in unit load warehouses to increase handling performance. International Journal of Production Economics, 170:838–849, 2015.

Y. Bukchin, E. Khmelnitsky, and P. Yakuel. Optimizing a dynamic order-picking process. European Journal of Operational Research, 219(2):335–346, 2012.

H. J. Carlo and G. E. Giraldo. Toward perpetually organized unit-load warehouses. Computers & Industrial Engineering, 63(4):1003–1012, 2012.

T. Chabot, L. C Coelho, J. Renaud, and Jean-François Côté. Mathematical model, heuristics and exact method for order picking in narrow aisles. Journal of the Operational Research Society, pages 1–12, 2017a.

T. Chabot, R. Lahyani, L. C. Coelho, and J. Renaud. Order picking problems under weight, fragility and category constraints. International Journal of Production Research, 55(21): 6361–6379, 2017b.

C. Chackelson, A. Errasti, D. Ciprés, and F. Lahoz. Evaluating order picking performance trade-offs by configuring main operating strategies in a retail distributor: a design of experiments approach. International Journal of Production Research, 51(20):6097–6109, 2013.

L. Chen, A. Langevin, and D. Riopel. A tabu search algorithm for the relocation problem in a warehousing system. International Journal of Production Economics, 129(1):147–156, 2011.

M.-C. Chen and H.-P. Wu. An association-based clustering approach to order batching considering customer demand patterns. Omega, 33(4):333–343, 2005.

E.-P. Chew and L. C. T. Travel time analysis for general item location assignment in a rectangular warehouse. European Journal of Operational Research, 112(3):582–597, 1999.

D. M.-H. Chiang, C.-P. Lin, and M.-C. Chen. The adaptive approach for storage assignment by mining data of warehouse management system for distribution centres. Enterprise Information Systems, 5(2):219–234, 2011.

N. Christofides and I. Colloff. The rearrangement of items in a warehouse. Operations Research, 21(2):577–589, 1973.

L. C. Coelho and G. Laporte. The exact solution of several classes of inventory-routing problems. Computers & Operations Research, 40(2):558–565, 2013.

J.-F. Cordeau and G. Laporte. The dial-a-ride problem: models and algorithms. Annals of Operations Research, 153(1):29–46, 2007.

G. Cormier and E. A. Gunn. A review of warehouse models. European Journal of Operational Research, 58(1):3–13, 1992.

H. Davarzani and A. Norrman. Toward a relevant agenda for warehousing research: literature review and practitioners' input. Logistics Research, 8(1):1–18, 2015.

R. B. M. De Koster. Warehouse assessment in a single tour. In Warehousing in the Global Supply Chain, pages 457–473. Springer, 2012.

R. B. M. De Koster and E. S. Van der Poort. Routing orderpickers in a warehouse: a comparison between optimal and heuristic solutions. IIE Transactions, 30(5):469–480, 1998.

R. B. M. De Koster, E. S. Van der Poort, and K. J. Roodbergen. When to apply optimal or heuristic routing of order pickers. In B. Fleischmann, J. A. E. E. Van Nunen, M.G. Speranza, and P. Stähly, editors, Advances in Distribution Logistics, pages 375–401. Springer, Berlin, 1998.

R. B. M. De Koster, K. J. Roodbergen, and R. Van Voorden. Reduction of walking time in the distribution center of de bijenkorf. In New Trends in Distribution Logistics, pages 215–234. Springer, 1999a.

R. B. M. De Koster, E. S. Van der Poort, and M. Wolters. Efficient order batching methods in warehouses. International Journal of Production Research, 37(7):1479–1504, 1999b.

R. B. M. De Koster, T. Le-Duc, and K. J. Roodbergen. Design and control of warehouse order picking: A literature review. European Journal of Operational Research, 182(2):481–501, 2007.

R. B. M. De Koster, A. L. Johnson, and D. Roy. Warehouse design and management. International Journal of Production Research, 55(21):6327–6330, 2017.

J. De Vries, R. B. M. De Koster, and D. Stam. Exploring the role of picker personality in predicting picking performance with pick by voice, pick to light and rf-terminal picking. International Journal of Production Research, 54(8):2260–2274, 2016.

R. Dekker, R. B. M. De Koster, K. J. Roodbergen, and H. Van Kalleveen. Improving order-picking response time at ankor's warehouse. Interfaces, 34(4):303–313, 2004.

J. Drury. Towards more efficient order picking, volume 1. 1988.

E. A. Elsayed and M.-K. Lee. Order processing in automated storage/retrieval systems with due dates. IIE Transactions, 28(7):567–577, 1996.

E. A. Elsayed, M.-K. Lee, S. Kim, and E. Scherer. Sequencing and batching procedures for minimizing earliness and tardiness penalty of order retrievals. International Journal of Production Research, 31(3):727–738, 1993.

E. Frazelle. World-Class Warehousing and Material Handling, volume 280. McGraw-Hill, New York, 2002.

E. H. Frazelle, S. T. Hackman, U. Passy, and L. K. Platzman. The Forward-Reserve Problem. John Wiley & Sons, Inc., New-York, 1994.

A. J. R. M. Gademann, J. P. Van Den Berg, and H. H. Van Der Hoff. An order batching algorithm for wave picking in a parallel-aisle warehouse. IIE Transactions, 33(5):385–398, 2001.

N. Gademann and S. Velde. Order batching to minimize total travel time in a parallel-aisle warehouse. IIE Transactions, 37(1):63–75, 2005.

M. Garfinkel. Minimizing multi-zone orders in the correlated storage assignment problem. PhD thesis, Georgia Tech., 2005.

D. R. Gibson and G. P. Sharp. Order batching procedures. European Journal of Operational Research, 58(1):57–67, 1992.

M. Goetschalckx and J. Ashayeri. Classification and design of order picking. Logistics World, 2(2):99–106, 1989.

M. Goetschalckx and D. H. Ratliff. Order picking in an aisle. IIE Transactions, 20(1):53–62, 1988.

Y. Gong and R. B. M. De Koster. A polling-based dynamic order picking system for online retailers. IIE Transactions, 40(11):1070–1082, 2008.

E. H. Grosse, C. H. Glock, and W. P. Neumann. Human factors in order picking: a content analysis of the literature. International Journal of Production Research, 55(5):1260–1276, 2017.

J. Gu, M. Goetschalckx, and L. F. McGinnis. Research on warehouse operation: A comprehensive review. European Journal of Operational Research, 177(1):1–21, 2007.

J. Gu, M. Goetschalckx, and L. F. McGinnis. Solving the forward-reserve allocation problem in warehouse order picking systems. Journal of the Operational Research Society, 61(6): 1013–1021, 2010a.

J. Gu, M. Goetschalckx, and L. F. McGinnis. Research on warehouse design and performance evaluation: A comprehensive review. European Journal of Operational Research, 203(3): 539–549, 2010b.

K. R. Gue and R. D. Meller. Aisle configurations for unit-load warehouses. IIE Transactions, 41(3):171–182, 2009.

K. R. Gue, R. D. Meller, and J. D. Skufca. The effects of pick density on order picking areas with narrow aisles. IIE Transactions, 38(10):859–868, 2006.

K. R. Gue, G. Ivanović, and R. D. Meller. A unit-load warehouse with multiple pickup and deposit points and non-traditional aisles. Transportation Research Part E: Logistics and Transportation Review, 48(4):795–806, 2012.

S. T. Hackman and L. K. Platzman. Near-optimal solution of generalized resource allocation problems with large capacities. Operations Research, 38(5):902–910, 1990.

S. T. Hackman, M. J. Rosenblatt, and J. M. Olin. Allocating items to an automated storage and retrieval system. IIE Transactions, 22(1):7–14, 1990.

R. W. Hall. Distance approximations for routing manual pickers in a warehouse. IIE Transactions, 25(4):76–87, 1993.

W. H. Hausman, L. B. Schwarz, and S. C. Graves. Optimal storage assignment in automatic warehousing systems. Management Science, 22(6):629–638, 1976.

S. Henn. Algorithms for on-line order batching in an order picking warehouse. Computers & Operations Research, 39(11):2549–2563, 2012.

S. Henn and V. Schmid. Metaheuristics for order batching and sequencing in manual order picking systems. Computers & Industrial Engineering, 66(2):338–351, 2013.

S. Henn and G. Wäscher. Tabu search heuristics for the order batching problem in manual order picking systems. European Journal of Operational Research, 222(3):484–494, 2012.

S. Henn, S. Koch, and G. Wäscher. Order batching in order picking warehouses: a survey of solution approaches. In R. Manzini, editor, Warehousing in the Global Supply Chain. Springer, London, 2012.

J. L. Heskett. Cube-per-order index - a key to warehouse stock location. Transportation and Distribution Management, 3(1):27–31, 1963.

Y.-C. Ho and Y.-Y. Tseng. A study on order-batching methods of order-picking in a distribution centre with two cross-aisles. International Journal of Production Research, 44(17): 3391–3417, 2006.

Y.-C. Ho, T.-S. Su, and Z.-B. Shi. Order-batching methods for an order-picking warehouse with two cross aisles. Computers & Industrial Engineering, 55(2):321–347, 2008.

M. Hompel and T. Schmidt. Warehouse Management: Automation and Organisation of Warehouse and Order Picking Systems. Springer Science & Business Media, Berlin, 2006.

S. Hong, A. L. Johnson, and B. A. Peters. Batch picking in narrow-aisle order picking systems with consideration for picker blocking. European Journal of Operational Research, 221(3): 557–570, 2012.

H. Hwang and M.-K. Lee. Order batching algorithms for a man-on-board automated storage and retrieval system. Engineering Costs and Production Economics, 13(4):285–294, 1988.

J. M Jarvis and E. D. McDowell. Optimal product layout in an order picking warehouse. IIE Transactions, 23(1):93–102, 1991.

M. E. Johnson and M. L. Brandeau. Stochastic modeling for automated material handling system design and control. Transportation Science, 30(4):330–350, 1996.

L. Junqueira, R. Morabito, and D. S. Yamashita. Three-dimensional container loading models with cargo stability and load bearing constraints. Computers & Operations Research, 39 (1):74–85, 2012.

I. Kara, G. Laporte, and T. Bektaş. A note on the lifted miller–tucker–zemlin subtour elimination constraints for the capacitated vehicle routing problem. European Journal of Operational Research, 158(3):793–795, 2004.

B.-I. Kim, S. S. Heragu, R. J. Graves, and A. St. Onge. Realization of a short cycle time in warehouse replenishment and order picking. International Journal of Production Research, 41(2):349–364, 2003.

S. Kirsner. Amazon buys warehouse robotics start-up kiva systems for $775 million, 2012. URL http://archive.boston.com/business/technology/innoeco/2012/03/amazon_buys_warehouse_robotics.html.

S. Koch and G. Wäscher. A grouping genetic algorithm for the order batching problem in distribution warehouses. Journal of Business Economics, 86(1-2):131–153, 2016.

M. Kofler, A. Beham, S. Wagner, M. Affenzeller, and W. Achleitner. Re-warehousing vs. healing: Strategies for warehouse storage location assignment. In 3rd IEEE International Symposium on Logistics and Industrial Informatics, pages 77–82. IEEE, 2011.

M. Kofler, A. Beham, S. Wagner, and M. Affenzeller. Affinity based slotting in warehouses with dynamic order patterns. In Advanced Methods and Applications in Computational Intelligence, pages 123–143. Springer, 2014.

R. Lahyani, F. Semet, and B. Trouillet. Vehicle routing problems with scheduling constraints. In Metaheuristics for Production Scheduling, pages 433–463. Wiley Online Library, United Kingdom, 2014.

R. Lahyani, L. C. Coelho, and J. Renaud. Alternative formulations and improved bounds for the multi-depot fleet size and mix vehicle routing. Technical Report CIRRELT-2015-36, Québec, Canada, 2015.

D. M. Lambert, J. R. Stock, and L. M. Ellram. Fundamentals of logistics management, 1998.

G. Laporte. Generalized subtour elimination constraints and connectivity constraints. Journal of the Operational Research Society, 37(5):509–514, 1986.

G. Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. European Journal of Operational Research, 59(2):231–247, 1992.

E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys. The traveling salesman problem. a guided tour of combinatorial optimisation, 1985.

T. Le-Duc. Design and control of efficient order picking processes. Erasmus Research Institute of Management (ERIM), 2005.

T. Le-Duc and R. B. M. De Koster. Travel distance estimation and storage zone optimization in a 2-block class-based storage strategy warehouse. International Journal of Production Research, 43(17):3561–3581, 2005.

T. Le-Duc and R. B. M. De Koster. Travel time estimation and order batching in a 2-block warehouse. European Journal of Operational Research, 176(1):374–388, 2007.

W. Lu, D. McFarlane, V. Giannikas, and Q. Zhang. An algorithm for dynamic order-picking in warehouse operations. European Journal of Operational Research, 248(1):107–122, 2016.

J. Lysgaard. CVRPSEP: A package of separation routines for the capacitated vehicle routing problem. Institut for Driftøkonomi og Logistik, Handelshøjskolen i Århus, 2003.

R. Manzini. Warehousing in the Global Supply Chain. Springer-Verlag, London, 2012.

M. Matusiak, R. B. M. De Koster, L. Kroon, and J. Saarinen. A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse. European Journal of Operational Research, 236(3):968–977, 2014a.

M. Matusiak, R. B. M. De Koster, L. Kroon, and J. Saarinen. A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse. European Journal of Operational Research, 236(3):968–977, 2014b.

M. Matusiak, R. B. M. De Koster, and J. Saarinen. Utilizing individual picker skills to improve order batching in a warehouse. European Journal of Operational Research, 263(3):888 – 899, 2017.

C. P. Medard and N. Sawhney. Airline crew scheduling from planning to operations. European Journal of Operational Research, 183(3):1013–1027, 2007.

R. D. Meller and K.-Y. Gau. The facility layout problem: recent and emerging trends and perspectives. Journal of Manufacturing Systems, 15(5):351–366, 1996.

C. H. Mowrey and P. J. Parikh. Mixed-width aisle configurations for order picking in distribution centers. European Journal of Operational Research, 232(1):87–97, 2014.

İ. Muter and T. Öncan. An exact solution approach for the order batching problem. IIE Transactions, 47(7):728–738, 2015.

M. M. Naim and J. Gosling. On leanness, agility and leagile supply chains. International Journal of Production Economics, 131(1):342–354, 2011.

T. Öncan. MILP formulations and an iterated local search algorithm with tabu thresholding for the order batching problem. European Journal of Operational Research, 243(1):142–155, 2015.

M. W. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. SIAM Review, 33(1):60–100, 1991.

C.-H. Pan and S.-Y. Liu. A comparative study of order batching algorithms. Omega, 23(6): 691–700, 1995.

C.-H. Pan, J, P.-H. Shih, and M.-H. Wu. Order batching in a pick-and-pass warehousing system with group genetic algorithm. Omega, 57:238–248, 2015.

J. C.-H. Pan, P.-H. Shih, and M.-H. Wu. Storage assignment problem with travel distance and blocking considerations for a picker-to-part order picking system. Computers & Industrial Engineering, 62(2):527–535, 2012.

P. J. Parikh and R. D. Meller. A note on worker blocking in narrow-aisle order picking systems when pick time is non-deterministic. IIE Transactions, 42(6):392–404, 2010.

B. C. Park. Order picking: issues, systems and models. In R. Manzini, editor, Warehousing in the Global Supply Chain, pages 1–30. Springer-Verlag, London, 2012.

J. Park and B.-I. Kim. The school bus routing problem: A review. European Journal of Operational Research, 202(2):311–319, 2010.

J. A. Pazour and H. J. Carlo. Warehouse reshuffling: Insights and optimization. Transportation Research Part E: Logistics and Transportation Review, 73:207–226, 2015.

A. Pessoa, E. Uchoa, and M. V. S. Poggi de Aragão. A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. Networks, 54(4):167–177, 2009.

C. G. Petersen. An evaluation of order picking routing policies. International Journal of Operations & Production Management, 17(11):1098–1111, 1997.

C. G. Petersen. The impact of routing and storage policies on warehouse efficiency. International Journal of Operations & Production Management, 19(10):1053–1064, 1999.

C. G Petersen. Considerations in order picking zone configuration. International Journal of Operations & Production Management, 22(7):793–805, 2002.

C. G. Petersen and G. R. Aase. A comparison of picking, storage, and routing policies in manual order picking. International Journal of Production Economics, 92(1):11–19, 2004.

C. G. Petersen and R. W. Schmenner. An evaluation of routing and volume-based storage policies in an order picking operation. Decision Sciences, 30(2):481, 1999.

C. G. Petersen, G. R. Aase, and D. R. Heiser. Improving order-picking performance through the implementation of class-based storage. International Journal of Physical Distribution & Logistics Management, 34(7):534–544, 2004.

J. C. Picard and M. Queyranne. The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. Operations Research, 26 (1):86–110, 1978.

M. V. S. Poggi de Aragão and E. Uchoa. New exact algorithms for the capacitated cehicle routing problem. In P. Toth and D. Vigo, editors, Vehicle Routing: Problems, Methods, and Applications, pages 59–86. MOS-SIAM Series on Optimization, Philadelphia, 2014.

J.-Y. Potvin and J.-M. Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. European Journal of Operational Research, 66 (3):331–340, 1993.

H. D. Ratliff and A. S. Rosenthal. Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. Operations Research, 31(3):507–521, 1983.

J. Renaud and A. Ruiz. Improving product location and order picking activities in a distribution centre. Journal of the Operational Research Society, 59(12):1603–1613, 2008.

K. J. Roodbergen. Layout and routing methods for warehouses. PhD thesis, Erasmus University, the Netherlands, 2001.

K. J. Roodbergen. Storage assignment for order picking in multiple-block warehouses. In R. Manzini, editor, Warehousing in the Global Supply Chain, pages 139–155. Springer, 2012.

K. J. Roodbergen and R. B. M. De Koster. Routing methods for warehouses with multiple cross aisles. International Journal of Production Research, 39(9):1865–1883, 2001a.

K. J. Roodbergen and R. B. M. De Koster. Routing order pickers in a warehouse with a middle aisle. European Journal of Operational Research, 133(1):32–43, 2001b.

K. J. Roodbergen and I. F. A. Vis. A model for warehouse layout. IIE Transactions, 38(10): 799–811, 2006.

K. J. Roodbergen and I. F. A. Vis. A survey of literature on automated storage and retrieval systems. European Journal of Operational Research, 194(2):343–362, 2009.

K. J. Roodbergen, G. P. Sharp, and I. F. A. Vis. Designing the layout structure of manual order picking areas in warehouses. IIE Transactions, 40(11):1032–1045, 2008.

K. J. Roodbergen, I. F. A. Vis, and G. D. Taylor Jr. Simultaneous determination of warehouse layout and control policies. International Journal of Production Research, 53(11):3306–3326, 2015.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science, 40(4):455–472, 2006.

M. B. Rosenwein. An application of cluster analysis to the problem of locating items within a warehouse. IIE Transactions, 26(1):101–103, 1994.

B. Rouwenhorst, B. Reuter, V. Stockrahm, G. J. Van Houtum, R. J. Mantel, and W. H. M. Zijm. Warehouse design and control: Framework and literature review. European Journal of Operational Research, 122(3):515–533, 2000.

D. Roy, A. Krishnamurthy, S. S. Heragu, and C. Malmborg. A multi-tier linking approach to analyze performance of autonomous vehicle-based storage and retrieval systems. Computers & Operations Research, 83:173–188, 2017.

R. A. Ruben and F. R. Jacobs. Batch construction heuristics and storage assignment strategies for walk/ride and pick systems. Management Science, 45(4):575–596, 1999.

B. R. Sarker and P. S. Babu. Travel time models in automated storage/retrieval systems: A critical review. International Journal of Production Economics, 40(2):173–184, 1995.

M. W. P. Savelsbergh and M. Sol. The general pickup and delivery problem. Transportation Science, 29(1):17–29, 1995.

A. Scholz and G. Wäscher. Order batching and picker routing in manual order picking systems: the benefits of integrated routing. Central European Journal of Operations Research, 25 (2):491–520, 2017.

A. Scholz, D. Schubert, and G. Wäscher. Order picking with multiple pickers and due dates–simultaneous solution of order batching, batch assignment and sequencing, and picker routing problems. European Journal of Operational Research, 2017.

A. H. Schrotenboer, S. Wruck, K. J. Roodbergen, M. Veenstra, and A. S. Dijkstra. Order picker routing with product returns and interaction delays. International Journal of Production Research, 55(21):6394–6406, 2017.

K. Selviaridis and M. Spring. Third party logistics: a literature review and research agenda. The International Journal of Logistics Management, 18(1):125–150, 2007.

F. Semet, P. Toth, and D. Vigo. Classical exact algorithms for capacited vehicle routing problem. In P. Toth and D. Vigo, editors, Vehicle Routing: Problems, Methods, and Applications, volume 18. SIAM, Philadelphia, 2014.

P. Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems. APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK, 1997.

N. Slack. Operations Strategy. Wiley Online Library, Harlow, U.K., 2015.

T. Sprock, S. Murrenhoff, and L. F. McGinnis. A hierarchical approach to warehouse design. International Journal of Production Research, 55(21):6331–6343, 2017.

L. C. Tang and E.-P. Chew. Order picking systems: batching and storage assignment strategies. Computers & Industrial Engineering, 33(3-4):817–820, 1997.

J. A. Tompkins, J. A. White, Y. A. Bozer, and J. M. A. Tanchoco. Facilities Planning. John Wiley & Sons, New York, 2010.

P. Toth and D. Vigo. Vehicle Routing: Problems, Methods, and Applications. MOS-SIAM Series on Optimization, Philadelphia, 2014.

C. A. Valle, J. E. Beasley, and A. S. da Cunha. Optimally solving the joint order batching and picker routing problem. European Journal of Operational Research, 262(3):817–834, 2017.

J. P. Van Den Berg. A literature survey on planning and control of warehousing systems. IIE Transactions, 31(8):751–762, 1999.

J. P. van den Berg. A literature survey on planning and control of warehousing systems. IIE Transactions, 31(8):751–762, 1999.

T. Van Gils, K. Ramaekers, A. Caris, and M. Cools. The use of time series forecasting in zone order picking systems to predict order pickers' workload. International Journal of Production Research, 55(21):6380–6393, 2017a.

T. Van Gils, K. Ramaekers, A. Caris, and R. B. M. De Koster. Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. European Journal of Operational Research, 2017b. URL http://dx.doi.org/10.1016/j.ejor.2017.09.002.

T. S. Vaughan and C. G. Petersen. The effect of warehouse cross aisles on order picking efficiency. International Journal of Production Research, 37(4):881–897, 1999.

R. Walter, N. Boysen, and A. Scholl. The discrete forward-reserve problem – allocating space, selecting products, and area sizing in forward order picking. European Journal of Operational Research, 229(3):585–594, 2013.

G. Wäscher. Order picking: a survey of planning problems and methods. In H. Dyckhoff, R. Lackes, and J. Reese, editors, The Vehicle Routing Problem, pages 323–347. Springer Verlag, Berlin, 2004.

S. Wruck, I. F. A. Vis, and J. Boter. Time-restricted batching models and solution approaches for integrated forward and return product flow handling in warehouses. Journal of the Operational Research Society, 64(10):1505–1516, 2013.

M. Yu and R. B. M. De Koster. Enhancing performance in order picking processes by dynamic storage systems. International Journal of Production Research, 48(16):4785–4806, 2010.

E. E. Zachariadis, C. D. Tarantilis, and C. T. Kiranoudis. The vehicle routing problem with simultaneous pick-ups and deliveries and two-dimensional loading constraints. European Journal of Operational Research, 251(2):369–386, 2016.

J. Zhang, X. Wang, F. T. S. Chan, and J. Ruan. On-line order batching and sequencing problem with multiple pickers: A hybrid rule-based algorithm. Applied Mathematical Modelling, 2016.

I. Žulj, S. Kramer, and M. Schneider. A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem. European Journal of Operational Research, 264(2):653–664, 2018.