Mathematical models for the warehouse reassignment problem

Thomas Chabot[a] Leandro C. Coelho[a,b] Jacques Renaud[a]
Kees Jan Roodbergen[c]

[a]CIRRELT and Université Laval, Canada

[b]Canada Research Chair in Integrated Logistics, Université Laval, Canada

[c]Faculty of Economics and Business, Groningen University, the Netherlands

September 14, 2023

Abstract

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 technology, in the demand and/or product offers. 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 We demonstrate how one can move from an out-of-date storage assignment to a better one, in a minimum of working time.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.

Keywords: warehousing, reassignment, material handling, optimization, exact method.

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 products 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]. The best storage assignment depends on the volume of the demand, the frequency with which the product is needed, the picking strategies and technology used, among others. Some of these elements are discussed next.

When setting up a new warehouse, one has the possibility to fully optimize the storage assignment along with other important factors such as technology used, order picking strategies, and sizing of zones. Order picking refers to the strategies used for picking the products once they are needed. For example, one common strategy is to have (human) pickers walk along the warehouse to pick the required products. The picker begins their tour at the order preparation location, picks up the items, and returns to the order preparation location later. This location is also called the Input/Output (I/O) point. This is called manual picking; several alternatives exist for the pickers to walk around the warehouse: they can traverse all the aisles completely, which is known as the S-shape picking; they can traverse only a part of the aisles in strategies known as the largest gap or the mid-point picking; or they can have their picking sequences fully optimized. The picking strategy is affected by the locations and the characteristics of the products, the technology used, the layout of the warehouse, among others [Chabot et al., 2018]. When placing the products in a warehouse to solve the storage assignment problem, considering the future picking strategies can be of great benefit [Chabot et al., 2017]. Combining storage location and order picking, Silva et al. [2020] show that savings ranging from about 27% to 62% compared to solutions from common storage policies, based on the solutions of their heuristic algorithms.

Other than picking, the layout of the warehouse also affects the decision of where to locate the products. A warehouse is often divided in zones: zone A contains the "best" locations, often closer to the I/O point, and is dedicated to the products with the highest demand; zone B contains locations that are not so close to the I/O and typically hold products of medium demand. Finally, zone C consists of the locations furthest away in the warehouse, and usually hold products with the lowest demand. Determining the sizes and positions of these zones is known as zone sizing. The zone sizing also impacts the performance of DCs; determining the right sizes for zones of a class-based storage system is also highly dependent on the layout and order picking strategy, among others. Silva et al. [2022] used machine learning tools to estimate appropriate zone sizes based on several features (layout characteristics, storage policy, and routing policy). Hence again, storage location appears as a key element that helps determine the overall efficiency of warehouses. As can be seen, determining the storage assignment is a complex task, that is influenced by many factors, and will influence the performance of several warehousing activities. If the products are assigned to "wrong" positions, they will cause longer picking routes, higher costs, delays, and will reduce warehouse performance.

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 presented in Carlo and Giraldo [2012]. When demand changes or products are naturally replaced, it may happen that they end up using "wrong" locations in the warehouse. For example, if a product was highly demanded during the summer, it is natural that it would be stored on zone A locations, close to the I/O point of the warehouse. However, during the winter, demand may decrease, but some products remain stored at their old location, using a prime space in the warehouse. It could be interesting for warehousing managers to relocate these products elsewhere, and to use this prime location to store a product that faces a high demand during that period. The frequency of products reassignment varies from a company to another, depending on the type of industry. In large warehouses containing thousands of products, work related to picking and replenishment has been observed to reduce by up to 15%, translating into half a million dollar savings per year [Trebilcock, 2011], and Werling et al. [2008] suggest updating the product assignment at least four times a year. 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* which is studied in this chapter. It is fully described as follows.

The warehouse reassignment problem involves optimizing the process of moving products to their new storage locations within a warehouse. This optimization aims to improve efficiency in tasks like order picking and replenishment. The key feature is that it allows for moving from one current allocation to another by strategically swapping products, minimizing disruptions and costs while maximizing overall efficiency. Factors such as demand fluctuations, product life cycles, and storage policies are considered in advance by determining which products should be relocated to improve order picking efficiency. The goal is to strike a balance between the time required for reassignment and the future gains in picking and handling efficiency.

Demand fluctuations and product life cycles should not be overlooked when planning the storage assignment. Storage revisions can mobilize substantial resources and disrupt order picking activities. Despite the clear benefit of a good assignment for the minimization of movements involved in order picking, the trade-off between reassignment time and future picking and handling gains can be hard to measure. This requires a good knowledge of products, sales, forecasts and available capacity in terms of time and equipment.

The storage strategy must be selected according to several warehouse design decisions, also known as the storage location assignment problem (SLAP). 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 location 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 tradeoff 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. The difference between healing and a full rearrangement of the warehouse is that it is possible that by changing the location of only a few products, a significant improvement in performance is achieved. Often times, one knows about the most critical products that need to have their locations changed. In identifying these products, one knows about locations that will become available, and also knows where to relocate these products to. In this chapter, we exploit this parsimonious relocation of products to find performance improvement for the warehouse, without disrupting the daily activities too much as is needed in a full rearrangement.

The action of repositioning some products in a warehouse, also known as reassignment or reshuffling, was initially studied by Christofides and Colloff [1973]. Knowing an optimal storage location assignment (or a desired one), one must consider the handling method to move all the involved products. Chen et al. [2011] present a tabu search heuristic to relocate products 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 products to be relocated to destinations are decision variables. A cycle is composed of two or more products that exchange their positions. 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.

The reshuffling of pallets within a warehouse was studied by Buckow and Knust [2023], who aimed at completing the reshuffle as fast as possible. While this option has the advantage to completely reset the products of a warehouse to their desired locations, it also imposes significant workload. Moreover, during this time, the warehouse cannot be used to satisfy customer demands.

A restoring policy was defined by Linn and Wysk [1990b,a] who use idle times between picking waves to reorganize the warehouse, bringing fast moving products close to the I/O point. Muralidharan et al. [1995] also study a class-based storage strategy in which the reassignment of products is performed heuristically. They assume a limited amount of time to perform the reassignment process. More importantly, they assume that the ending location of each product is already available. Finally, Carlo and Giraldo [2012] study a rearrange-while-working, in which unit-load warehouses can reassign a pallet to another location upon retrieval. A summary of the approaches mentioned for product reassignment in warehouses is presented in Table 1.

Table 1: Overview of approaches for product reassignment in warehouses

Strategy name	Full rearrangement	Healing	
Impact on daily operation	High	Medium	Low
Problem name	Storage Location Assignment Problem	Rearrange-while-Working Problem	Warehouse Reassignment Problem
Customer demand	Static	Dynamic	Dynamic
Typical objective	Minimize average order-picking time	Minimize total time of picking + relocations	Minimize time to perform relocations
Typical constraints	None	Picking jobs + desired final storage location assignment	Desired final storage location assignment
Literature	Hausman et al. [1976]; Silva et al. [2020]	Carlo and Giraldo [2012]	Christofides and Colloff [1973]; Pazour and Carlo [2015]; **this chapter**

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, and only do it heuristically. Christofides and Colloff [1973] 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. 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 reassignment process. They propose a mathematical formulation for the reassignment problem in cycles. More importantly, they do not allow for a product switch to occur, instead allowing for an intermediary drop to happen, which can significantly increase the overall distance.

There are major differences between our problem settings and those of Christofides and Colloff [1973], and of Pazour and Carlo [2015]. The most important difference is that they only allow to move a product to an empty shelf, whereas we allow a product switch incurring a time penalty. This swap was also allowed by Buckow and Knust [2023]. However, we are the first ones to handle this problem exactly, by means of new and efficient formulations.

In order to understand our context easily, it is important to properly define the term *product switch*. Let product A be at 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 move it towards its destination. By performing a series of product switches, it is possible to reposition a large number of products and effectively obtain a new storage location warehouse-wide. Figure 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. This can be achieved by making several product switches.

The main contributions of this chapter are the development of the first 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

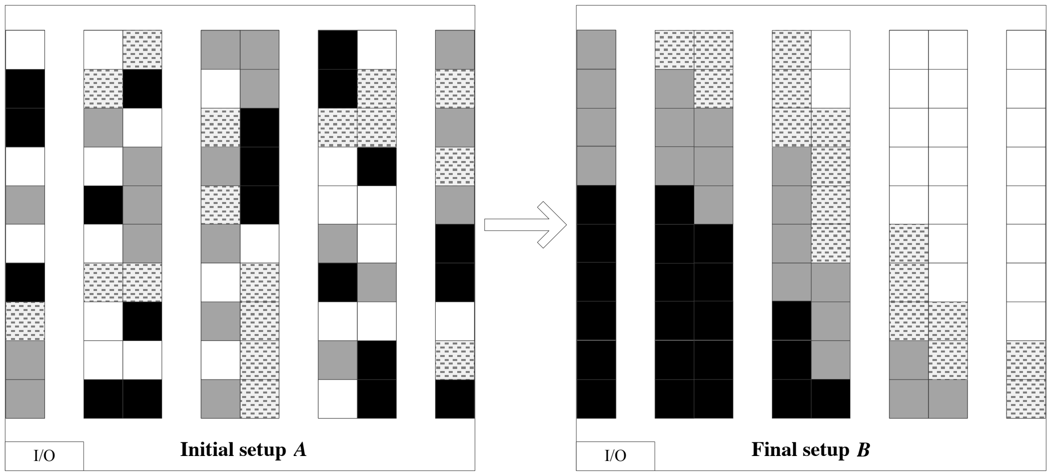

Figure 1: Reorganization of products inside the picking area, colored by picking frequency

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 2 presents the problem formulation and an illustrative example of the reassignment problem. Section 3 presents the mathematical models and a set of inequalities to strengthen the formulations. Section 4 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 5 concludes the chapter and presents some research perspectives.

2 Problem formulation

Following the formulation of Christofides and Colloff [1973] of the reassignment problem, let $A = [a_n]$, $n \in \{1, \dots, N\}$ 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 unit-load 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). Each 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 operators in such a way to minimize the total working time (traveling time, pick, drop and switch time).

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. The desired final assignment can be obtained using one of the methods described in the introduction by taking into account volumes, demand profiles, and picking strategies. It can consider, for example, that initially product a was the most requested product in the warehouse and has been located in the best available storage location; given demand fluctuations, product a is not required as much, and must have its location used by product e, which is now the most demanded product in the warehouse. In Table 2, 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 2 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 2: 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.

The assumptions of our model are simple and logical ones:

- the current and the desired assignments of the products are known,
- all vehicles performing the reassignments start and end their routes at the I/O point,
- the products being reassigned are moved directly from their current position to their new position, without intermediary drops,
- the goal is to finish all reassignments as soon as possible, thus minimizing the time needed for this activity.

The idea of reassigning the locations of some products in a warehouse is general and the example just described how it should work. This idea can be applied, e.g., in retail warehousing or e-commerce fulfillment centers, as they often need to reorganize products to accommodate seasonal demand changes, new product arrivals, clearance sales, changing product assortments, and order profiles. Efficiently relocating critical products can enhance order fulfillment and reduce costs, while optimizing the placement of fast-moving or frequently bundled products can improve order picking efficiency.

3 Mathematical models

This section presents three different mathematical formulations for the warehouse reassignment problem. Model 1 uses a uniquely designed graph with nodes and arcs that allow us to track the status of a location over time. While models 2 and 3 are traditional and widely used in the routing literature, model 3 is more compact as the status of the locations are preprocessed and only possible moves are modeled, based on a pickup and delivery formulation from the literature. The graph of Model 1 contains fewer nodes and arcs, leading to a model with significantly fewer variables and constraints, as will be demonstrated later. These models and particularly the new graph definition are described in the next sections.

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 3 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 3: Reassignment requests

i	1	2	3	4	5
$\mathcal{R}_i$	2	1	4	5	3

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 each of the following three models. Let the time limit of a route be $\mathcal{L}$.

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 an empty location or picking the product, represented by the sets $\mathcal{D}$ and $\mathcal{P}$. The third type of action is related to what happens after a drop in an 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. *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 2. This allows us to easily determine if a picker reaches a location with or without a product and if he leaves it with or without a product.

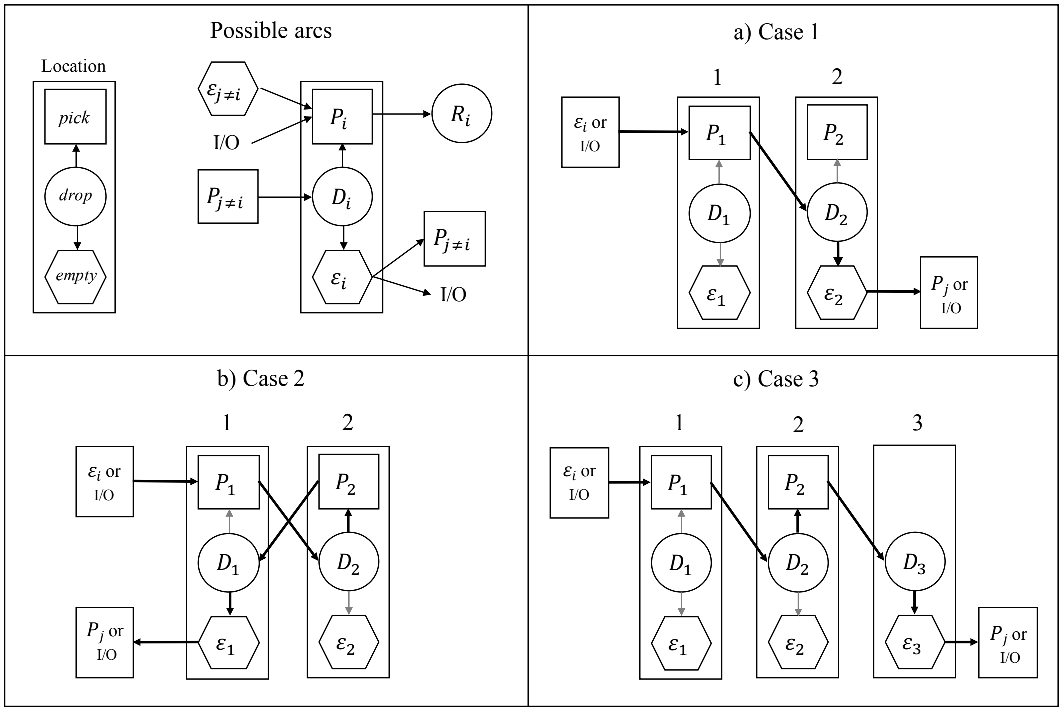

Figure 2: The three possible cases of sequences of nodes

Figure 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 2a), we reach pick $\mathcal{P}_1$ in location 1, complete 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 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 $\mathcal{P}_2$ (performing a product switch). 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 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:

$$x_{ij} \in E \; if \begin{cases} i = 0 \text{ and } j \in \mathcal{P} & \text{(1a)} \\ i = \mathcal{E}_n \text{ and } j \in \mathcal{P}_m \cup 0 \quad \forall \; n, m \in N : n \neq m & \text{(1b)} \\ i = \mathcal{D}_n \text{ and } j = \mathcal{P}_n \quad \forall \; n \in N & \text{(1c)} \\ i = \mathcal{D}_n \text{ and } j = \mathcal{E}_n \quad \forall \; n \in N. & \text{(1d)} \end{cases}$$

In equation (1a) we have all arcs between the I/O point (node 0) and all picks. In (1b), we have arcs from empty nodes in $\mathcal{E}$ to pick nodes of different locations or returning to the I/O point. Equation (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 (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 (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 presents a summary of the parameters, sets and variables used in our formulation.

Table 4: 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:

$$Min \; Z = \sum_{(i,j) \in E} c_{ij} \; x_{ij} + \sum_{i \in V'} w_i \tag{2}$$

subject to:

$$\sum_{i \in \mathcal{E}} x_{i0} = k, \tag{3}$$

$$\sum_{j\in\mathcal{P}} x_{0j} = k, \tag{4}$$

$$\sum_{i\in V\backslash\{\mathcal{P}\}} x_{ij} = 1 \qquad \forall j \in \mathcal{P}, \tag{5}$$

$$\sum_{\substack{i\in V\\ |(i,j)\in E}} x_{ij} = \sum_{\substack{k\in V\\ |(j,k)\in E}} x_{jk} \qquad \forall j \in V', \tag{6}$$

$$u_i - u_j + \mathcal{L}x_{ij} \leq \mathcal{L} - c_{ij} \qquad \forall (i,j) \in E, \tag{7}$$

$$u_{\mathcal{P}_n} \leq u_{\mathcal{D}_n} + (1 - x_{\mathcal{D}_n,\mathcal{E}_n})\mathcal{L} \qquad \forall n \in N : \mathcal{P}_n \notin \emptyset, \tag{8}$$

$$u_j \leq u_i + w_j + c_{ij} + (1 - x_{ij})\mathcal{L} \qquad \forall (i,j) \in E, \tag{9}$$

$$x_{\mathcal{P}_n,\mathcal{R}_n} = 1 \qquad \forall n \in N, \tag{10}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \tag{11}$$

$$0 \leq u_i \leq \mathcal{L} \qquad \forall i \in V, \tag{12}$$

$$0 \leq w_i \leq \mathcal{L} \qquad \forall i \in V, \tag{13}$$

$$k \in \mathbb{N}. \tag{14}$$

The objective (2) minimizes the total workload corresponding to the sum of the traveling time (including penalties), the service time, and the idle time. Constraints (3) and (4) fix the number of routes that respectively exit and enter the I/O point. Constraints (5) ensure that all pick nodes will be visited and in the same way, performing all reassignment requests. Constraints (6) ensure the flow equilibrium at each node. Constraints (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 (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 (9) are used to fix the idle time variable w_j if an arc x_{ij} is used. Constraints (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 (11) set the nature of variable x_{ij}. Constraints (12) and (13) bound the variable u_i and w_i, respectively, to a maximal value $\mathcal{L}$. Constraint (14) indicates that variable k is a positive integer.

3.2 Vehicle-indexed formulation (M2)

In this section we present a vehicle-indexed adaptation of the previous formulation which is based on the same graph definition. We use a set $\mathcal{M}$ of 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 picker (vehicle) $k \in \mathcal{M}$ travels between i and j, zero otherwise. Let us 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 (1a) and (1b) for three index variables x_{ij}^k on set E are updated as follows:

$$x_{ij}^k \in E \ \mathit{if} \begin{cases} i \in \mathcal{M}^+,\ j \in \mathcal{P} \cup \mathcal{M}^- & \forall\ k \in \mathcal{M} \quad (15a) \\ i = \mathcal{E}_n,\ j \in \mathcal{P}_m \cup \mathcal{M}^- & \forall\ n, m \in N : n \neq m, k \in \mathcal{M} \quad (15b) \end{cases}$$

The model is the following:

$$Min\ Z = \sum_{k\in\mathcal{M}} (u_{k^-} - u_{k^+}) \tag{16}$$

subject to:

$$\sum_{i\in\mathcal{E}\cup\{k^+\}} x^k_{ik^-} = 1, \qquad \forall k \in \mathcal{M}, \tag{17}$$

$$\sum_{j\in\mathcal{P}\cup\{k^-\}} x^k_{k^+j} = 1, \qquad \forall k \in \mathcal{M}, \tag{18}$$

$$\sum_{k\in\mathcal{M}} \sum_{i\in V'\cup\mathcal{M}^+\setminus\{\mathcal{P}\}} x^k_{ij} = 1 \qquad \forall j \in \mathcal{P}, \tag{19}$$

$$\sum_{\substack{i\in V'\cup\mathcal{M}^+\\|(i,j)\in E}} x^k_{ij} = \sum_{\substack{l\in V'\cup\mathcal{M}^-\\|(j,l)\in E}} x^k_{jl} \qquad \forall j \in V^{'}, k \in \mathcal{M}, \tag{20}$$

$$u_i - u_j + \mathcal{L}x^k_{ij} \leq \mathcal{L} - c_{ij} \qquad \forall (i,j,k) \in E, \tag{21}$$

$$u_{\mathcal{P}_n} \leq u_{\mathcal{D}_n} + (1 - \sum_{k\in\mathcal{M}} x^k_{\mathcal{D}_n,\mathcal{E}_n})\mathcal{L} \qquad \forall n \in N : \mathcal{P}_n \notin \emptyset, \tag{22}$$

$$x^k_{ij} \in \{0,1\} \qquad \forall (i,j,k) \in E, \tag{23}$$

$$0 \leq u_i \leq \mathcal{L} \qquad \forall i \in V. \tag{24}$$

The objective (16) minimizes the total workload by taking the difference between the ending and starting time for all vehicles. Constraints (17) and (18) make sure that each vehicle starts and ends at the I/O point. Constraints (19) ensure that each pick node will be visited only once. Constraints (20) ensure the flow equilibrium at a node. Constraints (21) allow the chronometer increment of variable u_i. Constraints (22) ensure the pick to be visited before the empty node of the same location when applicable. Constraints (23) and (24) define the nature of variables x^k_{ij} and u_i respectively.

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^k_i 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^k_{ij} 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:

$$Min\ Z = \sum_{k\in\mathcal{M}} (u_{k^-} - u_{k^+}) \tag{25}$$

subject to:

$$\sum_{k\in\mathcal{M}} z^k_i = 1 \qquad \forall i \in \mathcal{R}, \tag{26}$$

$$\sum_{j\in V} x^k_{lj} = \sum_{j\in V} x^k_{jl} = z^k_i \qquad \forall n \in N, l \in \mathcal{P}_n \cup \mathcal{R}_n, k \in \mathcal{M}, \tag{27}$$

$$\sum_{j\in V'\cup\{k^-\}} x^k_{k^+j} = 1 \qquad \forall k \in \mathcal{M}, \tag{28}$$

$$\sum_{j\in V'\cup\{k^+\}} x^k_{ik^-} = 1 \qquad \forall k \in \mathcal{M}, \tag{29}$$

$$u_p \leq u_d \qquad \forall n \in \mathcal{N}, p \in \mathcal{P}_n, d \in \mathcal{R}_n, \tag{30}$$

$$u_i - u_j + \mathcal{L}x_{ij}^k \leq \mathcal{L} - c_{ij} \qquad \forall (i,j) \in E, k \in \mathcal{M}, \tag{31}$$

$$y_{k^+} = 0 \qquad \forall k \in \mathcal{M}, \tag{32}$$

$$y_i \leq \sum_{k \in \mathcal{M}} z_i^k \qquad \forall i \in \mathcal{P}, \tag{33}$$

$$y_i - y_j + x_{ij}^k \leq 0 \qquad \forall i,j \in V, k \in \mathcal{M}, \tag{34}$$

$$u_{\mathcal{D}_n} - u_{\mathcal{P}_n} - (1 - \sum_{k \in \mathcal{M}} x^k_{\mathcal{D}_n \mathcal{P}_n})\mathcal{L} \leq c_{\mathcal{D}_n \mathcal{P}_n} \qquad \forall n \in N : p_n \neq \emptyset, \tag{35}$$

$$u_{\mathcal{D}_n} - u_{\mathcal{P}_n} + \sum_{k \in \mathcal{M}} x^k_{\mathcal{D}_n \mathcal{P}_n}\mathcal{L} \geq c_{\mathcal{D}_n \mathcal{P}_n} \qquad \forall n \in N : p_n \neq \emptyset, \tag{36}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall i,j \in V \cup W, k \in \mathcal{M}, \tag{37}$$

$$z_i^k \in \{0,1\} \qquad \forall i \in \mathcal{P}, k \in \mathcal{M}, \tag{38}$$

$$u_i \geq 0 \qquad \forall i \in V \cup W, \tag{39}$$

$$y_i \geq 0 \qquad \forall i \in V \cup W. \tag{40}$$

The objective function (25) minimizes the total workload for all vehicles. Constraints (26) ensure that all pick requests will be served only once. With constraints (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 (28) and (29), we make sure that each vehicle starts and ends at the I/O point. Constraints (30) ensure that the pick is made before the drop of the same request. Constraints (31) ensure the correct increment of the departure time variables when an arc is used. Constraints (32) and (33) impose the initial and maximum load of the vehicle respectively. Constraints (34) make the correct increment of the load when applicable. Together, constraints (35) and (36) ensure that product switch at the same location will be done if a drop is made at a still occupied location. Constraints (37) to (40) define the nature of all involved variables.

3.4 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 (41) and (42) are valid for all locations $n \in N$ such that $p = \mathcal{P}_n$, $d = \mathcal{R}_n$.

$$c_{0p} \leq u_p \leq \mathcal{L} - c_{pd} - c_{d0} \tag{41}$$

$$c_{0p} + c_{pd} \leq u_d \leq \mathcal{L} - c_{d0}. \tag{42}$$

The Miller-Tucker-Zemlin constraints (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:

$$u_i - u_j + (\mathcal{L} - min_k\{c_{ki}\})x_{ij} \leq \mathcal{L} - min_k\{c_{ki}\} - c_{ij} \quad \forall (i,j) \in E. \tag{43}$$

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

$$Z \geq + \sum_{n \in N} c_{\mathcal{P}_n \mathcal{D}_n} + a \left(\min_{p \in \mathcal{P}} \{c_{0p}\} + \min_{d \in \mathcal{D} \cup \mathcal{E}} \{c_{d0}\} \right) \tag{44}$$

$$a = \left\lceil \frac{\sum_{n \in N} c_{\mathcal{P}_n \mathcal{D}_n}}{\mathcal{L}} \right\rceil . \tag{45}$$

Equation (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 (41) – (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:

$$\sum_{j \in P \cup \{k^-\}} x_{0j}^k \leq \sum_{j \in P \cup \{k^-\}} x_{0j}^{k-1} \quad \forall k \in \mathcal{M} \backslash \{1\} \tag{46}$$

$$\sum_{\substack{l \in V \\ |(l,i) \in E}} x_{li}^k \leq \sum_{\substack{c \in V \\ |(c,j) \in E}} \sum_{j<i} x_{cj}^{k-1} \quad \forall i \in V', k \in \mathcal{M} \backslash \{1\}. \tag{47}$$

Constraints (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 (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 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 4.1. In Section 4.2 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 4.3. Finally, Section 4.4 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.

4.1 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. 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 $\mathcal{L} = \{\infty, 1000\}$. For a product switch, 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.

4.2 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).

4.3 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 5 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. We also observe that all three formulations clearly outperformed the CSH method, yielding solutions that take less than half the time needed by CSH.

Table 6 presents the upper bound and lower bound of all three models without any timing valid inequalities (41) and (42), initial lower bound (44) and symmetry breaking inequalities (46) and (47). We see that M1 still found 20 out of 30 optimal solutions and gives very similar bounds. However, the gap increases from 2.9% to 5.0%. 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 in Section 3.4.

The graph of Figure 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 5. 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 presents the number of constraints for each formulation. For instances with 9 requests and more,

Table 5: 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

Table 6: Model performances without inequalities and initial lower bound

Instances				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	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

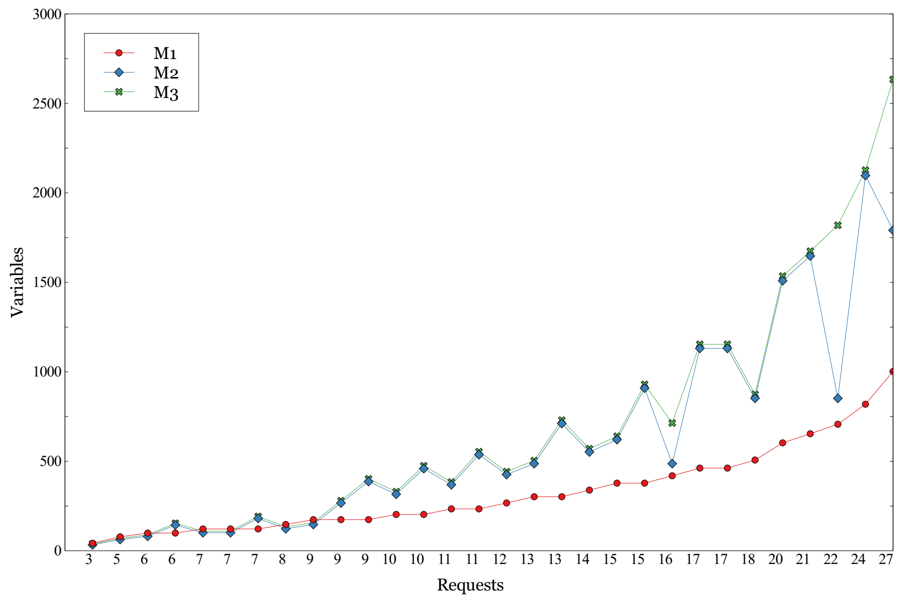

Figure 3: Number of variables

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 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. Formulations 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.

4.4 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 perform a 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

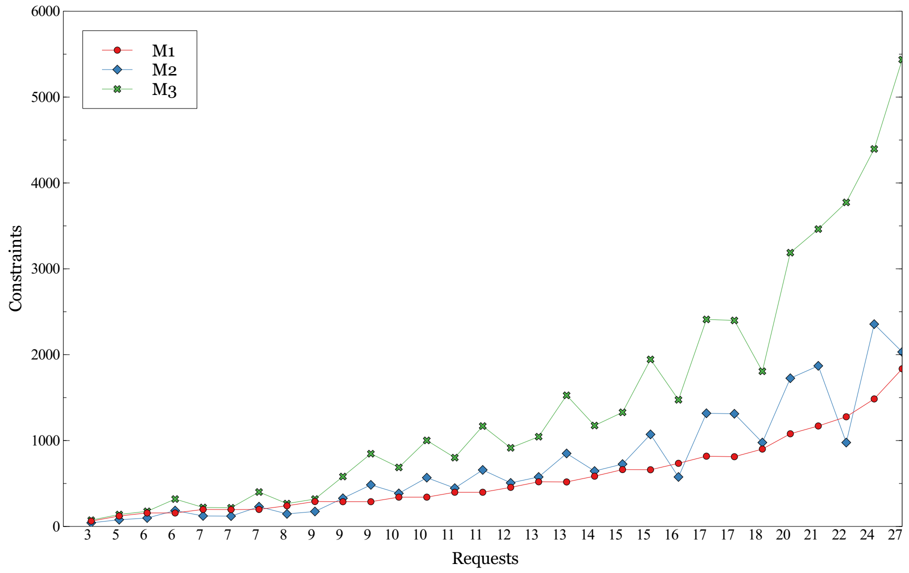

Figure 4: Number of constraints

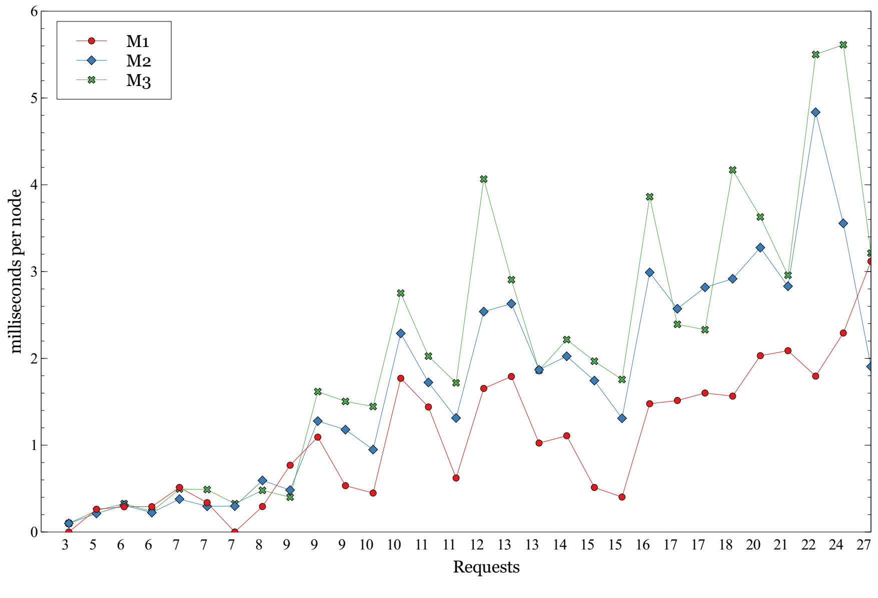

Figure 5: Milliseconds passed per visited nodes

row of Table 5. Figure 6 presents the results of our simulation. It shows the total distance of the unit-load picking with and without the reassignment. Table 5 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.

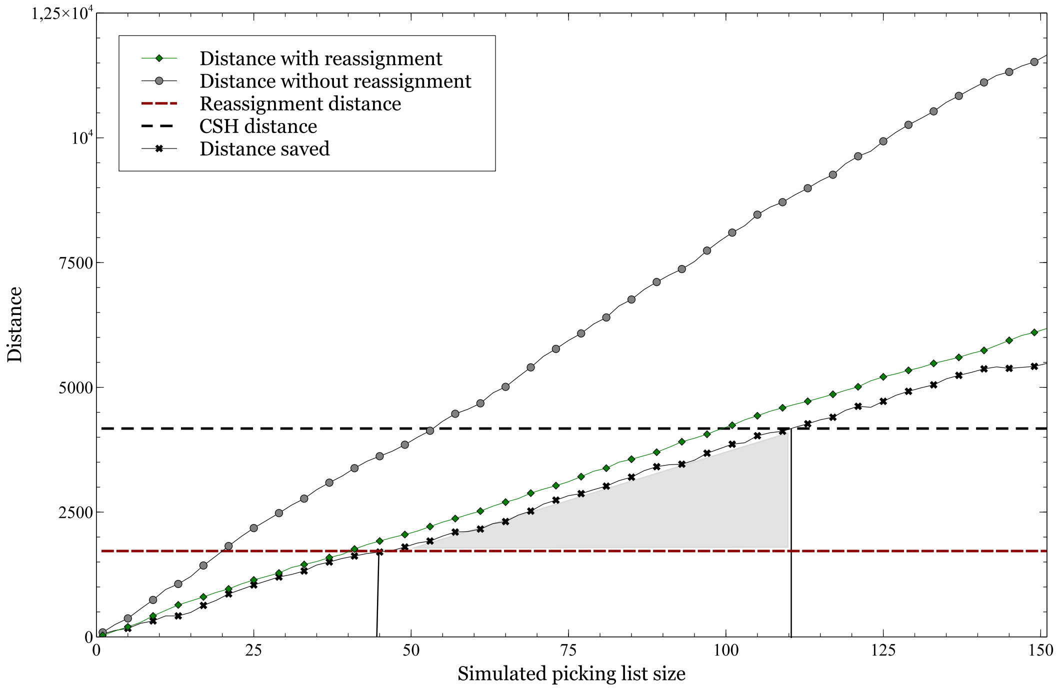

Figure 6: Simulation of picking scenarios

5 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 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.

Acknowledgments

This research was partly supported by the Canadian Natural Sciences and Engineering Research Council (NSERC) under grants 2019-00094 (L.C. Coelho) and 2023-04167 (J. Renaud), and by the Dutch Research Council (NWO), the Ministry of Infrastructure and Water Management, and the Dutch Institute for Advanced Logistics (TKI Dinalog) under grant 439.18.459 (K.J. Roodbergen). This support is gratefully acknowledged.

References

J.-N. Buckow and S. Knust. The warehouse reshuffling problem with swap moves. Transportation Research Part E: Logistics and Transportation Review, 169:102994, 2023.

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

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, 2017.

T. Chabot, L. C. Coelho, J. Renaud, and J.-F. Côté. Mathematical model, heuristics and exact method for order picking in narrow aisles. Journal of the Operational Research Society, 69(8):1242–1253, 2018.

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.

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.

R. 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.

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.

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. 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.

I. Kara, G. Laporte, and T. Bektas. 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.

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. J. Linn and R. A. Wysk. An expert system framework for automated storage and retrieval system control. Computers & Industrial Engineering, 18(1):37–48, 1990a.

R. J. Linn and R. A. Wysk. An expert system based controller for an automated storage/retrieval system. The International Journal of Production Research, 28(4):735–756, 1990b.

B. Muralidharan, R. J. Linn, and R. Pandit. Shuffling heuristics for the storage location assignment in an AS/RS. The International Journal of Production Research, 33(6):1661–1672, 1995.

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

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.

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

A. Silva, L. C. Coelho, M. Darvish, and J. Renaud. Integrating storage location and order picking problems in warehouse planning. Transportation Research Part E: Logistics and Transportation Review, 140:102003, 2020.

A. Silva, K. J. Roodbergen, L. C. Coelho, and M. Darvish. Estimating optimal ABC zone sizes in manual warehouses. International Journal of Production Economics, 252:108579, 2022.

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

B. Trebilcock. Should you reslot your warehouse? Technical report, Modern Materials Handling, May 2011. URL `https://www.mmh.com/article/resolve_to_reslot_your_warehouse`.

C. Werling, R. C. Daniell, T. Singer, and I. Hobkirk. The importance of slotting. Technical report, Warehousing Education and Research Council, February 2008. URL `http://www.werc.org/assets/1/Publications/Importance%20of%20Slotting%20WERCSheet_Feb08.pdf`.