Assigning and sequencing storage locations under a two level storage policy: optimization model and matheuristic approaches*

Giacomo Lanza† Mauro Passacantando‡ Maria Grazia Scutellà§

Abstract: We deal with a problem which combines storage location assignment with sequencing decisions about the assigned storage locations. Given a set of different product types, with the corresponding storage demand, a set of capacitated storage locations has to be assigned to each product type for the corresponding storing operations. In addition, a suitable sequencing of the assigned storage locations must be devised for each product type, i.e., it has to be decided the ordering with which the storage locations will be filled up during the storing operations. A motivation is that a First-In First-Out (FIFO) picking criterion among storage locations is required per product type. The sequencing established for the assigned storage locations will therefore allow to easily implement the FIFO policy in the successive order picking. Moreover, the selected sequencing also determines the availability of extra storage per product type, on top of pairs of consecutive storage locations along the sequence. The goal is to maximize the storage capacity which remains available after the assignment of the storage locations.

We prove the NP-Hardness of the problem and we model it as a constrained multicommodity flow problem on an auxiliary graph. We then propose a Mixed-Integer Linear Programming (MILP) model, with some valid inequalities, based on the multicommodity flow formulation. Two relaxations are proposed as well to estimate the quality of the model solutions. Two matheuristic approaches are then designed starting from the MILP model. The proposed methodology is applied to a case study related to a large warehouse with a high stock rotation index in tissue logistics, which motivated our study. Computational results on a wide test bed related to such a real application context show the efficiency and the efficacy of the presented approaches.

Keywords: Storage Location Assignment, Storage Location Sequencing, Mixed-Integer Linear Programming, Multicommodity Flows, Matheuristic.

*This paper has been published in Omega, vol. 108, Article 102565, see: `https://www.sciencedirect.com/science/article/pii/S0305048321001742?via%3Dihub`.

†Corresponding author; Department of Computer Science, University of Pisa, Largo Bruno Pontecorvo, 3, 56127, Pisa, Italy, email: `giacomo.lanza@di.unipi.it`

‡Department of Computer Science, University of Pisa, Pisa, Italy, email: `mauro.passacantando@unipi.it`

§Department of Computer Science, University of Pisa, Pisa, Italy, email: `scut@di.unipi.it`

1 Introduction

Warehouses are an essential component of any supply chain. Their operation systems are configured through the following basic processes: reception and dispatch, order picking, storage (Gu et al., 2007; Rouwenhorst et al., 2000). Reception and dispatch are the interface of a warehouse for incoming and outgoing material flow, and concern the organization of all the operations required to manage entering and exiting items, i.e., units of the managed product types. Units of the managed product types are often referred to as Stock Keeping Units (SKUs) in the related literature. Order picking is generally recognized as the most expensive warehouse operation, as it tends to be either very labor intensive or very capital intensive. It requires the organization of the orders to be picked up and of the resources needed for picking. Storage is concerned with the organization of the products held in the warehouse in order to achieve high space utilization and facilitate efficient material handling. The broad organization of the products in a warehouse is normally a strategic/tactical decision made on management (such as a dedicated storage area for a specific product type) or material handling considerations (such as a forward area for fast picking). This process results into a long-term *storage assignment policy*, which further defines the internal configuration of the warehouse (such as dimension of specific storage areas and aisle configurations) and that fixes the rules to follow when stocking of products is needed.

There are various storage assignment policies described in the literature (Gu et al., 2010; Roodbergen and Vis, 2009). The most representatives are the random, the dedicated, and the class-based allocation policies (see the pioneering studies of Hausman et al., 1976; Ashayeri and Gelders, 1985). The random policy involves the random assignment of the incoming items to any available and eligible location within the storage area, each location having an equal probability of being selected. In the dedicated assignment policy, the warehouse is divided into a number of zones, each of those dedicated to one product type. Replenishment of items of a certain product type always occurs at the corresponding dedicated zone. In the class-based policy, product types are classified into a number of classes based on their properties (such as demand rate, order frequency, dimension, or product correlations), each product class having a reserved zone within the warehouse. Accordingly, an incoming item of a product class is stored at an arbitrary available location within its reserved zone.

Once a long-term storage assignment policy has been defined, it is necessary to assign each incoming item to a storage location of the warehouse, possibly subject to additional rules depending on the specific application context. Examples of storage locations are shelves, cells or stacks. Thus, a storage location can store several items depending on its own capacity. The associated problem is generally known as the *Storage Location Assignment Problem* (SLAP). Specifically, given information on the availability, position and capacity of the storage locations, and given information on the set of items to be stored (such as product type and physical dimension), SLAP aims at determining the storage locations where items have to be

allocated in the warehouse by optimizing criteria such as material handling cost or storage space utilization, while respecting the storage assignment policy chosen at a strategic/tactical level and specific assignment rules, if present. SLAP is an operational decision problem, having a strong influence on other decisions within the warehouse, such as order picking and routing.

In this paper, we address a problem which combines storage location assignment with sequencing decisions about the assigned storage locations, and which originates from a real-world application context in tissue logistics. Specifically, given a set of different product types, each with its own storage demand expressed in number of items to store, a set of storage locations has to be assigned to each product type for the corresponding storing operations. Each storage location has a capacity which depends on the product type, i.e., a maximum number of items can be stored for that product type, and it can be assigned to at most one product type, i.e., different types of products cannot share the same storage location. In addition, a suitable sequencing of the assigned storage locations must be devised for each product type, i.e., it has to be decided the ordering with which the storage locations will be filled up during the storing operations. A motivation is that an order picking based on the time of permanence of the items in the warehouse has to be pursued. More precisely, a FIFO picking criterion among storage locations is required per product type. That is, separately per product type, items stored in a certain storage location cannot be retrieved if items in previously replenished storage locations have still to be picked up. Notice that the FIFO picking criterion is not required inside storage locations: in that case, the order of picking will depend on the specificity of the considered storage locations. The sequencing established for the assigned storage locations will thus have an impact on the successive order picking operations, by allowing to easily manage the required FIFO criterion. Moreover, for each product type, the selected sequencing also determines the availability of additional extra storage for that product type. Specifically, an additional amount of storage can be made available on the top of pairs of consecutive storage locations along the sequence, provided that they are fully replenished and physically contiguous. The amount of the available storage on the top does depend on the two storage locations at the ground level and on the product type to be stored. Notice that optimizing assignment and sequencing in such a way to be able to exploit extra top storage is particularly relevant for a clever storage management, especially in the case of warehouses usually near to their level of saturation, such as the ones characterized by a high index of product rotation. Additional soft constraints are present in the addressed problem, as better clarified next, with the goal of maximizing the storage capacity which remains available after the assignment of the storage locations.

We prove the NP-Hardness of the considered problem. Moreover, after its formulation in terms of constrained multicommodity flows on an auxiliary graph, we propose a Mixed-Integer Linear Programming (MILP) model, with some valid inequalities, based on the suggested multicommodity flow formulation. A strengthening of the MILP model for the special case of

acyclic auxiliary graph is also presented. Two alternative relaxations are then proposed to estimate the quality of the model solutions, which are based on storage location assignment and on aggregation of product types, respectively. Since the problem can be very hard to address computationally, two matheuristic approaches are designed starting from the proposed MILP model. The first approach rapidly constructs a solution of good quality via problem decomposition, and then provides it to the MILP model as a starting solution, in order to improve it. The second approach, instead, first assigns storage locations to the incoming items via the proposed assignment based relaxation, and then determines a sequencing of the assigned storage locations, solving a restricted version of the model itself. A case study is then presented, which is related to the tissue logistics sector and which motivated our research in this topic. The involved warehouse is larger than 10,000 m^2 and is characterized by a high product rotation index (specifically, more than 1,000 pallets are moved per day). Its modernization is the goal of a big research project funded by Regione Toscana, in Italy, and it includes the resolution of the considered combined assignment-sequencing storage location problem via Operations Research techniques. Computational experiments on real data provided by the company show the efficiency and the efficacy of the proposed approaches.

The paper is organized as follows. Section 2 reviews the main results from the literature. Specifically, the main results in the area of SLAPs are presented in Section 2.1. Since storage locations are stacks in the presented case study, storage systems based on stacks are reviewed in Section 2.2, as well as near systems such as the deep-lane ones. Multi-level storage assignment features are also discussed. Moreover, works taking into account picking aspects when assigning storage locations are presented in Section 2.3, then positioning our contribution with respect to the literature in Section 2.4. Section 3 describes the problem addressed in this paper in more detail. Section 4 presents the proposed MILP formulation and provides the corresponding NP-Hardness result. Valid inequalities, a strengthened model tailored to the considered case study, and some model relaxations are also presented. The matheuristic approaches built to tackle the problem are described in Section 5. Section 6 presents the case study and describes the experimental plan, by reporting the results of the computational experiments we performed. Finally, Section 7 concludes the paper and identifies some future directions of research.

2 Literature review

Storage assignment problems, also referred to as loading problems in a broader perspective (Lehnfeld and Knust, 2014), deal with the storage of incoming items. Each item reaching the storage area, which can be a warehouse, but at a more general level could be a yard, the bunt of a container ship or even a tram/bus depot, has to be assigned to a feasible location and stored until it is required to be retrieved. A storage assignment plan decides on

the exact storage position of each item in the storage system (Lehnfeld and Knust, 2014). As indicated before, such decisions are made by considering some long-term storage assignment policy (random, dedicated and class-based policy, previously mentioned, are the most popular), that broadly prescribes the rules to follow when stocking is needed, and which strictly depends on the considered application. An overview of container assignment policies in terminals can be found in Dekker et al. (2007) and Stahlbock and Voß (2008). Focusing on traditional (aisle-based) warehouses instead, an overview of warehouse management policies can be found in Rouwenhorst et al. (2000); De Koster et al. (2007) and Gu et al. (2007). Operationally, besides the chosen storage assignment policy, storage decisions are made by considering criteria related to material handling costs, stocking/retrieving efforts, time for order preparations, resource utilization for stocking/picking operations, warehouse space utilization or even energy consumption.

Since storage assignment problems originate from a wide range of different applications, a lot of scientific literature exists dealing with problems motivated from practice. These problems are often described with different terms and notions depending on the characteristics of the storage system and on the specific context, as overviewed next.

2.1 Storage Location Assignment

When items have to be stored on racks or shelves accessible from the side, then the problem is commonly referred to as the *Storage Location Assignment Problem*, or SLAP, as previously introduced (Rouwenhorst et al., 2000; De Koster et al., 2007; Gu et al., 2007). The SLAP literature is quite diversified because of the variety of peculiar storage assignment policies, (Li et al., 2016; Pang and Chan, 2017; Trindade et al., 2021), storage system customization (Bortolini et al., 2015; Hübner and Schaal, 2017; Bianchi-Aguiar et al., 2018; Ostermeier et al., 2021), product management rules (Zhang et al., 2017), internal layouts of warehouses (Ramtin and Pazour, 2015; Foroughi et al., 2020) and optimization criteria (Ang et al., 2012; Meneghetti and Monti, 2014; Yang et al., 2015; Battini et al., 2016; Ene et al., 2016; Larco et al., 2017) requested in real situations. We mention here Quintanilla et al. (2015), who consider a real problem raised by a Spanish company, where pallets of different weights need to be stocked in a chaotic warehouse. The warehouse has a rectangular layout with a certain number of racks on both sides of a set of parallel storage aisles. Racks may accommodate items on a double-level (one on top of the other) and on double-depth (one behind another). However, heaviest pallets should be stored below a maximum level and the most fragile ones cannot have another pallet on top of them. The crucial point to manage for the company is the remaining available storage capacity after the allocation of the items. The authors propose a MILP model with constraints on positions and level restrictions, whose objective is maximizing the space available for future assignments after the current SLAP is solved. Several heuristic algorithms and a local search procedure are presented.

2.2 Stacking, Deep-Lane and Multi-level Storage Assignment

Some storage systems consist of multiple stacks where items are stored on top of (e.g., a pile of containers) or back-to-back to (e.g., a queue of pallets) each other. These systems are typically considered when dealing with the storage of high volumes of incoming items having large inventory and high turnover (Accorsi et al., 2017). When items are piled one on top of another on vertical stacks, and cranes are employed to move the items and can only access the topmost ones, as in case of containers or steel slabs in yards, the problem is usually referred to as a *Stacking problem* (Lehnfeld and Knust, 2014). On the other hand, when items are stored on horizontal stacks on the ground, such as items stored back-to-back of each other on queue accessible only frontally, then the problem is known as a *Deep-Lane Storage Assignment problem* (Boysen et al., 2018). The two problems are extremely similar since the horizontal height of stacks in the first problem corresponds to the vertical depth of deep-lanes in the second one.

Considering the first kind of problems, container stacking problems have been widely studied and reviews may be found in Vis and De Koster (2003); Dekker et al. (2007); Stahlbock and Voß (2008) and Carlo et al. (2014). They may raise in storage yards, where containers are stored temporarily after they are discharged from vessels or before they are loaded onto vessels, or in vessels themselves, where containers are stowed and additional ship stability constraints are imposed.

On the other hand, focusing on deep-lane storage systems, in different applications incoming items may be not only stored back-to-back of each other, but also one on top of each other. The so-called stackability in a deep-lane context, defines the number of levels a deep-lane may have to store items by still maintaining the storage safety (Accorsi et al., 2017). In particular, Zaerpour et al. (2015) consider a multi-level deep-lane storage system where unit loads are stored on deep-racks following a shared storage policy, i.e. different product types share the same lane. Moreover, the assignment of items to a deep-lane belonging to a certain level may be done no matter if the lower levels are already occupied.

The multi-level feature is particularly addressed in the steel industry, where coils can usually be stored one on top of the others up to two levels for stability and safety. At the ground level a coil may be stored at any available position in the storage area, while at the upper level a coil can only be stored on top of two adjacent ground-level coils in the same row. Thus, coils are triangularly related, while normally items on stacks are vertically related (Tang et al., 2012). In Zäpfel and Wasner (2006) the storage assignment of steel coils considering such a storage policy is accounted for. Coils are of different types and the only restriction to exploit the triangular storage policy is that a coil can be positioned in an upper storage location if the location is empty and both locations underneath contain coils, no matter of the product type. A unique overhead traveling crane moves coils to perform storage operations simultaneously with retrieval and reshuffling operations. The problem is compared

to a job shop scheduling problem and formulated as a nonlinear integer programming model aiming at minimizing the completion time for the last order during the planning period. A local search based heuristic is proposed and tested through computation. Tang et al. (2015) consider the storage assignment of coils onto the stowage of a ship. The authors formulate the problem as a MILP aiming at minimizing a combination of ship instability throughout the entire voyage, the shuffles needed for unloading at the destination ports, and the dispersion of coils in the stowage destined to the same port. A construction heuristic coupled with a tabu search algorithm is developed, and several valid inequalities are proposed to help reducing the solution time. Other applications where this special stacking structure is considered are Tang et al. (2012, 2014); Xie et al. (2014) and Maschietto et al. (2017), where the focus however is the reassignment of locations for those coils blocking other ones targeted to retrieve.

2.3 Storage locations assignment addressing retrieval order aspects

Frequently, when stacks are filled up with not interchangeable items, some specific retrieval orders, such as the Last-in First-Out (LIFO) or the FIFO one, are requested to be followed when picking items to fulfill the orders requested (Gu et al., 2007). In particular, the FIFO policy is often taken into account, especially for those products with a peculiar product life (like fresh or frozen food). Even though this is a common practice in many real situations, the required retrieval order is normally addressed only when picker routing is planned, and often only approximations are used (Pang and Chan, 2017; Accorsi et al., 2017). Instead, no specific actions are usually taken when storage locations are assigned to incoming items to ease guiding retrieval operations later on, when orders are to be satisfied.

To the best of the author's knowledge, very few contributions deal with this aspect. Some of them have already been reviewed before, focusing however on the kind of storage system used rather than on picking order considerations. These approaches address the problem of storing items in such a way as not to block other retrieving items, or at least to minimize blocking situations, where an item is said to be blocked if one or more items with later retrieval times are stocked above/in front of it. To retrieve the current target item thus, other ones need to be removed from the current stack and located into other positions through premashalling and reshuffling operations (Carlo et al., 2014; Lehnfeld and Knust, 2014; Maniezzo et al., 2021).

Revillot-Narváez et al. (2020) consider a compact storage system composed of multi-level deep-racks accessible from the front, where items of different product types share the same lane. When pallets are stored, an ascending ranking is reported on their tags, and their retrieval has to be performed by strictly following this FIFO order. The authors develop two ILP models (with and without premarshalling) to assign locations to items aiming at minimizing the number of reshuffles over the planning horizon. Two greedy-randomised heuristic

approaches are proposed to solve large real-size instances, which are tested on realistic data from a frozen food distribution centre in Chile.

Slightly different are the approaches of Zaerpour et al. (2015); Boysen et al. (2018) and Boywitz and Boysen (2018), that consider a specific procedure to store items grouped in orders into a set of deep lanes. In particular, to enable a smooth retrieval process, orders are stored based on the arrival time (or time windows) of the truck in which they will be loaded. No two pallets destined to different trucks with overlapping retrieval intervals are assigned to the same lane. Zaerpour et al. (2015) consider multi-level deep-lane racks with frontal and posterior access, where a FIFO retrieval policy is followed. Earlier orders are positioned first, while later orders as last in the system. The authors propose a MILP formulation whose objective is to minimize the total retrieval time, and a three step constructive heuristic to solve real instances. Boysen et al. (2018) consider a problem in distribution centers handling fresh produce. A deep-lane refrigerated storage system is considered, all lanes having identical capacity and being initially empty. Pallets of food assembled during the day according to the demands of supermarkets have to be intermediately stored until the next morning, when trucks servicing the supermarkets have to be loaded. A storage assignment is sought such that blockings are avoided and the minimum number of lanes is utilized. The authors investigate the most usual case in which deep-lanes allow only a front access compared to a novel system in which access is allowed from both sides. In the first case, a LIFO policy for retrieval needs to be considered: later orders are positioned on the back while earlier ones at the front of the system. In the second case, a FIFO policy for retrieval is applied. The authors propose two MILP models and provide a simple yet effective solution procedure based on a problem decomposition. The computation of robust solutions against unpunctual arrival times of trucks is then addressed in Boywitz and Boysen (2018) for the front access deep-lane system.

2.4 Positioning our problem with respect to the literature

The problem presented in this paper shares some features with storage assignment problems from the literature. Nevertheless, they have never been considered jointly in a unique setting. Firstly, for each product type, storage location assignment decisions are taken simultaneously with sequencing decisions about the order of using the assigned storage locations. The aim is twofold: *i*) to enable a FIFO retrieval policy among storage locations later on, separately per product type, and *ii*) to exploit additional storage availability on the top of pairs of consecutive storage locations along each sequence. To the best of our knowledge, the combination of such features has never been addressed before in the storage assignment literature.

Moreover, the considered two-level storage policy is different than others from the literature. By considering in particular the triangular policy addressed in Zäpfel and Wasner (2006), reviewed in Section 2.2, there the only condition to locate an item at the upper level is that the lower locations are already occupied. Consequently, the number of positions po-

tentially available at the upper level is fixed and only depends on the number of locations on the ground level. In our problem, instead, an additional condition has to be respected: both the storage locations in each sequence (at the lower level) and the storage locations made available on the top of consecutive storage locations along the sequence must be assigned to the same product type. Therefore, the number of potentially available storage locations at the upper level is not fixed, rather it may change depending on the amount of items to store and their product type. This also affects the calculation of the storage remaining available after the item allocation.

3 The Sequence-based 2-Level SLAP

The problem is defined in a warehouse composed of a set $\mathcal{D}$ of departments, each having a given number of storage locations. Let $\mathcal{K}$ denote the set of the different product types requiring storage, and let q^k be the number of items of product type k that need storage, for each $k \in \mathcal{K}$. A storage location can store only items of the same product type, and the number of items that can be stored, also referred to as the capacity of the storage location, does depend on the product type. Precisely, if items of type k are assigned to a storage location i, then its capacity is c_i^k, meaning that at most c_i^k items of type k can be stored in i.

The majority of the product types can be stored in any available storage location, i.e., a random storage policy is considered. However, special product types do exist, which have to be preferably managed according to a dedicated storage policy. Precisely, we consider n disjoint subsets $\mathcal{K}_1^s, \ldots, \mathcal{K}_n^s$ of special product types, that should be preferably stored in departments belonging to the subsets $\mathcal{D}_1^s, \ldots, \mathcal{D}_n^s$, respectively. Notice that $\mathcal{D}_1^s, \ldots, \mathcal{D}_n^s$ are not necessarily disjoint.

As previously described, in addition to assign a set of storage locations to each product type $k \in \mathcal{K}$, a suitable sequencing of the assigned storage locations must be devised for each product type. That is, for each product type it has to be decided the ordering with which the assigned storage locations will be filled up during the storing operations. The motivation is twofold. Firstly, the selected sequencing enables a FIFO picking criterion among storage locations, separately per product type, as required in the kind of warehouses under consideration. Once a sequencing has been established, in fact, it will be sufficient to follow the corresponding ordering among storage locations during the successive picking steps in order to guarantee the satisfaction of the FIFO criterion. Notice that the satisfaction of the FIFO picking criterion among items stored in the same storage location is not required instead, the picking criterion in that case strongly depending on the characteristics of the used storage locations. The FIFO retrieval policy is typical of warehouses in application contexts such as the pharmaceutical and the tissue ones. The latter is the one addressed in the presented case study, in Section 6.1.

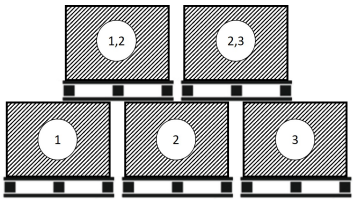


Figure 1: Sequence-based two-level storage policy representation.

Secondly, for each product type, the selected sequencing has an influence on the availability of extra storage for that product type. Specifically, additional storage can be made available on the top of each pair of consecutive storage locations along the sequence, provided that they are physically contiguous. The requirement that the two storage locations must also be consecutive along the sequence is due to the requested FIFO picking criterion among storage locations: if two storage locations, say i and j, would not be consecutive along the sequence selected for a certain product type k, and therefore would not be consecutively replenished during the stocking operations of k, then the items put on their top could block items having a very different time of permanence in the warehouse, by making difficult the corresponding FIFO picking realization. On the contrary, the considered policy enables an easy implementation of the FIFO picking criterion among storage locations, at least at the ground level: if the storage location j immediately follows the storage location i along the sequence of a certain product type k, and they are physically contiguous, during the storing operations of k store items on top of i and j immediately after the full replenishment of i (before) and of j (next); then, during the picking operations, firstly retrieve from the top of i and j, so allowing to pick items, at the ground level, firstly from i and then from j, thus guaranteeing the FIFO property among the storage locations at the ground level.

Hereafter this policy will be referred to as *sequence-based two-level storage policy*. An example is shown in Figure 1. In the figure, three storage locations are assigned to a certain product type at the ground level, and they are depicted from the front. The associated number represents their ordering in the corresponding sequence: 1, 2 and 3. By assuming that the storage location 1 is physically contiguous to the storage location 2, and 2 is physically contiguous to 3, then an extra storage is made available on top of 1 and 2, denoted by (1,2) in the figure. Similarly, an extra storage is made available on top of 2 and 3, denoted by (2,3). Accordingly, storing operations will be performed by firstly replenishing storage location 1, then storage location 2, and after storing on their top (i.e., at (1,2)). The storing operations will proceed at storage location 3, then storing on the top of 2 and 3, i.e., at (2,3). In order to guarantee the FIFO property at the ground level, picking operations will be performed starting from the top storage (1,2), and then picking items at the ground level from storage location 1. After, items from the top storage (2,3) will be retrieved, by thus allowing to retrieve items before from storage location 2 and then from storage location 3.

Indeed, the sequencing decisions rely on a set $\mathcal{P}$, containing the pairs of storage locations which can be consecutive in a sequence. $\mathcal{P}$ is usually defined according to some *Quality of Service* considerations, and it expresses some logical conditions about the order of replenishment of the storage locations. For example, if two storage locations, say i and j, are too far, the management might not want that j be replenished immediately after i. According to this, a storage location j can be the successor of a storage location i in a sequence if and only if $(i, j) \in \mathcal{P}$. For each $(i, j) \in \mathcal{P}$ and for each product type k, the amount of extra storage which can be made available on the top of i and j will be denoted by b_{ij}^k. Notice that this amount generally does depend on the two storage locations at the ground level and on the product type to be stored. In particular, $b_{ij}^k = 0$ for each k if i and j are not physically contiguous.

By summarizing, the addressed problem consists in assigning a sequence of available storage locations to each product type $k \in \mathcal{K}$, by satisfying the following constraints:

- each storage location can be assigned to a unique product type,
- the sum of the capacities of the storage locations assigned to a product type k, plus the extra storage made available on the top level thanks to the sequence-based two-level storage policy, must be greater than or equal to the storage demand q^k,
- the subsets $\mathcal{K}_1^s, \ldots, \mathcal{K}_n^s$ of special product types should be preferably stored in departments belonging to the subsets $\mathcal{D}_1^s, \ldots, \mathcal{D}_n^s$, respectively,

while maximizing the residual storage capacity, i.e., the storage capacity which remains available in the warehouse after the assignment of the storage locations. This problem will be named S2L-SLAP, which stands for Sequence-based 2-Level SLAP.

4 A multicommodity flow model

In this section, we propose a multicommodity flow formulation to S2L-SLAP, with some valid inequalities. We also present two relaxations, and prove the NP-Hardness of S2L-SLAP.

4.1 The auxiliary graph

In order to formulate the problem, we introduce an auxiliary graph $G = (N, A)$ describing the availability of storage locations in the warehouse.

The set of nodes N consists of: (i) a set $\mathcal{S}$ containing one node for each storage location available in the warehouse; (ii) a fictitious source node Σ; (iii) a fictitious sink node Θ. A balance c_i^k is associated with each node $i \in \mathcal{S}$ for each $k \in \mathcal{K}$, which gives the capacity (i.e., the availability in terms of items) of the corresponding storage location for product type k.

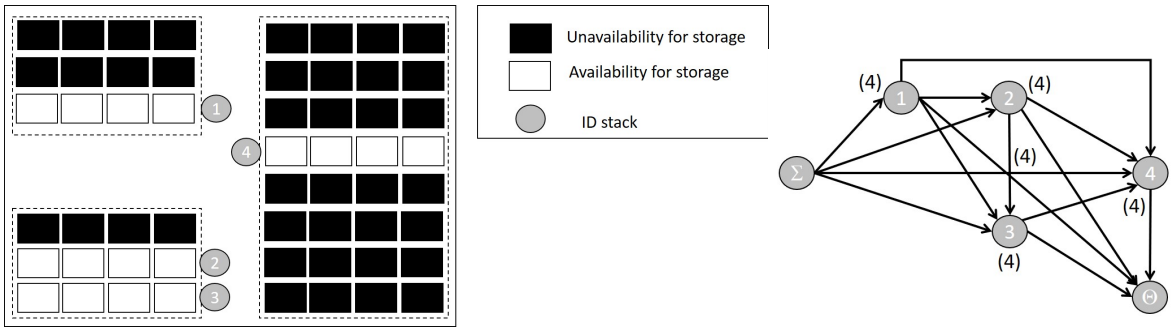
The set of arcs A defines the pairs of storage locations which can be consecutive in a sequence. As indicated before, the modeling idea is to associate a suitable sequence of available

storage locations with each $k \in \mathcal{K}$, thus specifying in which order the storage locations have to be used when storing and, consequently, the order with which storage locations will be emptied for shipping operations later on. Specifically, the set of arcs A consists of: (i) arcs (Σ, j), with $j \in \mathcal{S}$, to model the assignment of the first storage location to a product type; (ii) arcs (i, j), with $i, j \in \mathcal{S}$ and $(i, j) \in \mathcal{P}$, to model the assignment of the available storage location j immediately after the available storage location i (recall that $\mathcal{P}$ defines the allowed set of consecutive storage locations along a sequence); (iii) arcs (i, Θ), with $i \in \mathcal{S}$, to model the assignment of the last storage location to a product type. In particular, the subset of arcs connecting two contiguous storage locations will be denoted by

$$A_c = \{(i, j) \in A : \quad \text{storage locations } i \text{ and } j \text{ are contiguous}\}.$$

A weight b_{ij}^k, associated with each arc $(i, j) \in A$ for each product type $k \in \mathcal{K}$, indicates the amount of extra storage which can be made available on top of i and j for product type k. Notice that $b_{ij}^k = 0$ for each $k \in \mathcal{K}$ if $(i, j) \notin A_c$.

Example 4.1 *Consider the warehouse in Figure 2a. At the ground level there are six storage locations on the left and eight storage locations on the right. Each storage location has a capacity four, independently of the product type, and it is depicted as a row of four rectangles, one for each item that can be stored. The rectangles are full black colored to denote unavailability (i.e., items of some product types have already been stored) or white to denote availability for storage. The available storage locations in the warehouse are four and they are outlined in the figure by an identifier inside a gray circle. Since storage locations 2 and 3 are contiguous, an additional storage can be made available on top of them, say four additional items. Figure 2b reports the auxiliary graph associated with the considered instance. As in the presented case study, the set $\mathcal{P}$ contains each pair of available storage locations (i, j) with $i < j$. The nodes other than Σ and Θ, one for each available storage location, are marked with the same identifiers used in Figure 2a, and the corresponding balances, i.e., the capacities of the available storage locations, are associated between brackets. Regarding the arc weights,*



(a) Warehouse configuration (b) Graph representation

Figure 2: Warehouse configuration and related graph representation.

a weight 4 is associated with the arc (2,3), to represent the additional storage that can be exploited on top of the contiguous storage locations 2 and 3, while the weight is 0 for all the other arcs (weights equal to 0 are omitted in the figure).

4.2 Sequence of assigned storage locations as a directed path

We model the sequence of storage locations to assign to each product type $k \in \mathcal{K}$ by means of a directed path in the auxiliary graph from node Σ to node Θ, along which the quantity q^k to be stocked in the warehouse is sent, thus modeling the storing operations at the selected storage locations. This is formulated in terms of a flow pushed along the path. However, this is a non-traditional flow pushing. In fact, the amount of flow of product type k entering a node j along the path is suitably decreased, according to the capacity of the involved storage location, in order to model a storing operation at that storage location. Specifically, the flow is decreased by c_j^k, unless the predecessor of j along the path, say i, represents a storage location which is contiguous to j. In the latter case, the flow of product type k is decreased by $c_j^k + b_{ij}^k$, because of the sequence-based two-level storage policy previously described. A progressive flow reduction so gives rise for each $k \in \mathcal{K}$.

Example 4.1 (continued). *Considering again the Example 4.1, suppose that 8 items of product type 1 and 10 items of product type 2 need to be stored. A possible solution is given in Figure 3. The two paths, corresponding to the product types 1 and 2, respectively, are drawn with different dashed lines. The flow along each path is also reported, modeling the storing operations along the corresponding storage locations. Considering the product type 1, a flow of value 8 is pushed along the corresponding path, since 8 items need to be stored. The first storing operation is performed at storage location 1. This storage location is saturated, i.e., 4 items are stored, and the remaining 4 items move along the path to be stocked in storage location 4, which is the second in the sequence. As for product type 2, a flow equal to 10 is pushed along the corresponding path, due to its storage request. The first assigned storage location is 2, where 4 items are stocked. The remaining 6 items are then moved to the second*

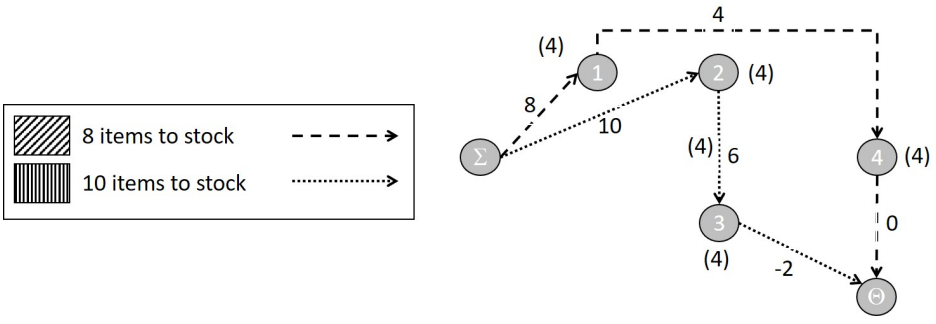
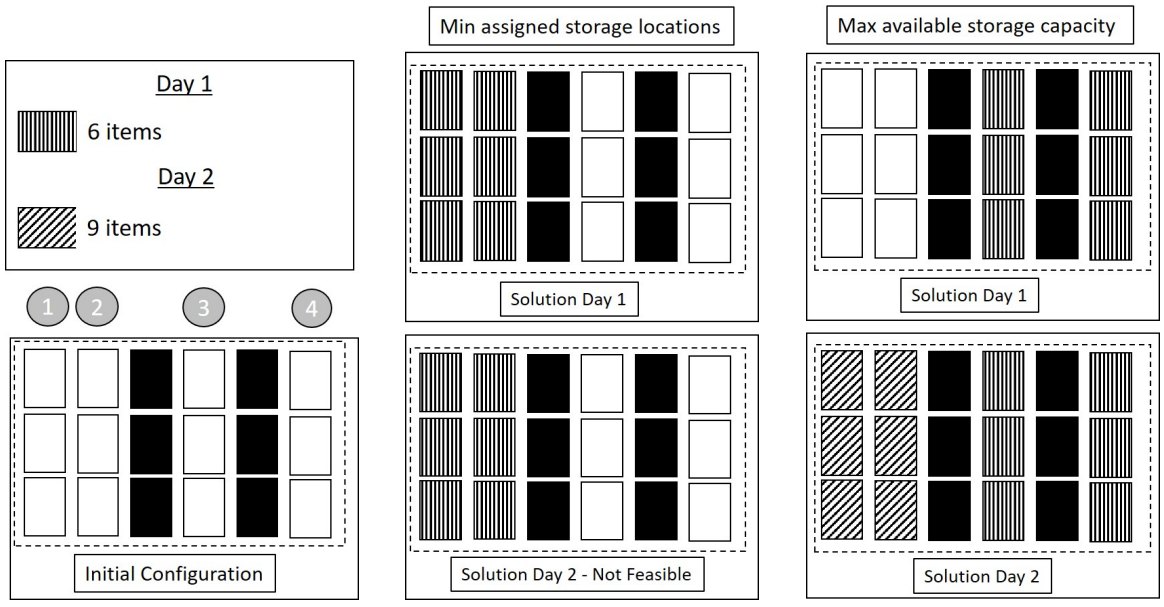


Figure 3: Solution representation in terms of directed paths.

assigned storage location, i.e., 3, in terms of a flow equal to 6. Since storage locations 2 and 3 are physically contiguous (see Figure 2a), the extra storage on their top can be exploited, and the 6 items are stocked in storage location 3 (4 items) and on the top of storage locations 2 and 3 (2 items). Notice that a negative flow, equal to -2, is associated with the last arc of the path. This value represents the opposite of the residual storage availability on top of storage locations 2 and 3: that is, 4 (extra storage on top of 2 and 3) - 2 (number of items stored on top of 2 and 3) = 2.

4.3 The objective function

The goal of S2L-SLAP is to maximize the storage capacity available after the assignment of the storage locations, which may be a very critical objective in several application contexts. Maximizing the available storage capacity is not equivalent to minimize the number of assigned storage locations, due to the sequence-based two-level storage policy. Figure 4 highlights the difference between the two objectives. Consider two consecutive days in which, respectively, 6 and 9 items of two different product types need to be stored, and the initial configuration of the warehouse reported in Figure 4a. As in the previous example, black colored rectangles denote unavailability, whereas white colored rectangles denote availability. Figure 4b shows an optimal solution in case the minimization of the assigned storage locations is the objective of the problem. The items needing storage in Day 1 are stocked into storage locations 1



(a) Items to store and warehouse configuration.

(b) Minimizing assigned storage locations.

(c) Maximizing storage capacity available after assignment.

Figure 4: Difference between minimizing assigned storage locations (b) and maximizing available storage capacity after assignment (c).

and 2; as a result, there is no enough space available for storing items of Day 2. Figure 4c describes instead the unique optimal solution in case the available room in the warehouse after the assignment is maximized, under the assumption that the extra storage available on top of the contiguous storage locations 1 and 2 is equal to 3. The items needing storage in Day 1 are stocked into the available storage locations 3 and 4. The available storage after this assignment is thus 9, i.e., the two contiguous storage locations 1 and 2, each with capacity 3, plus the extra storage equal to 3 on top of them. The 9 items needing storage in Day 2 can then be stocked, i.e., no infeasibility is now achieved.

4.4 Model description

To formulate S2L-SLAP, we introduce three families of variables:

1. *Assignment variables:*
   - $z_i^k \in \{0,1\}$, for any $i \in \mathcal{S}$ and $k \in \mathcal{K}$, model the assignment of storage location i to product type k;
   - $z_i^a \in \{0,1\}$, for any $i \in \mathcal{S}$, model the storage locations that are available after the assignment;
   - $z_{ij}^a \in \{0,1\}$, for any $(i,j) \in A_c$, model the extra storage, on top of the contiguous storage locations i and j, which is available after the assignment;
2. *Sequencing variables:* $x_{ij}^k \in \{0,1\}$, for any $(i,j) \in A$ and $k \in \mathcal{K}$, model the directed path selected for product type k in terms of a unitary flow from node Σ to node Θ;
3. *Flow variables:* $f_{ij}^k \in \mathbb{R}$, for any $(i,j) \in A$ and $k \in \mathcal{K}$, model the storing operations along the sequence of storage locations assigned to k in terms of a suitable flow along the corresponding path.

Using these variables, the proposed multicommodity flow model can be stated as follows:

$$\max \sum_{i \in \mathcal{S}} c_i^{max} z_i^a + \sum_{(i,j) \in A_c} b_{ij}^{max} z_{ij}^a + w \sum_{k \in \bigcup_{p=1}^{n} \mathcal{K}_p^s} \sum_{i \in \mathcal{S}} \delta_i^k z_i^k \tag{1}$$

subject to:

$$z_i^a + \sum_{k \in \mathcal{K}} z_i^k = 1 \qquad \forall\, i \in \mathcal{S}, \quad (2)$$

$$z_{ij}^a \leq z_i^a \qquad \forall\, (i,j) \in A_c, \quad (3)$$

$$z_{ij}^a \leq z_j^a \qquad \forall\, (i,j) \in A_c, \quad (4)$$

$$\sum_{j \in N^-(i)} x^k_{ji} - \sum_{j \in N^+(i)} x^k_{ij} = \begin{cases} -1 & \text{if } i = \Sigma \\ 0 & \text{if } i \in \mathcal{S} \\ 1 & \text{if } i = \Theta \end{cases} \qquad \forall\ k \in \mathcal{K},\ \forall\ i \in N, \quad (5)$$

$$z^k_i = \sum_{j \in N^-(i)} x^k_{ji} \qquad \forall\ k \in \mathcal{K},\ \forall\ i \in \mathcal{S}, \quad (6)$$

$$\sum_{j \in N^+(\Sigma)} f^k_{\Sigma j} = q^k \qquad \forall\ k \in \mathcal{K}, \quad (7)$$

$$\sum_{j \in N^-(i)} f^k_{ji} - \sum_{j \in N^+(i)} f^k_{ij} = \sum_{j \in N^-(i)} c^k_i x^k_{ji} + \sum_{\substack{j \in N^-(i): \\ (j,i) \in A_c}} b^k_{ji} x^k_{ji} \qquad \forall k \in \mathcal{K},\ \forall\ i \in \mathcal{S}, \quad (8)$$

$$0 \leq f^k_{ij} \leq q^k x^k_{ij} \qquad \forall\ k \in \mathcal{K}, \forall\ (i,j) \in A:\ j \neq \Theta, \quad (9)$$

$$-2c^{max}\, x^k_{i\Theta} \leq f^k_{i\Theta} \leq 0 \qquad \forall\ k \in \mathcal{K},\ \forall\ (i,\Theta) \in A. \quad (10)$$

The objective function (1) consists of two parts. The first two summations define the primary optimization goal, i.e., the storage capacity available after the assignment, to be maximized, while the last sum aims to satisfy the secondary optimization goal, i.e., the request that special product types should be preferably stored in specific departments, which is thus pursued as a soft constraint. Specifically, the first sum equals to the capacity of the storage locations which are available after the assignment. Here c^{max}_i does denote the maximum among the c^k_i capacities of storage location i. The second sum counts the extra storage which is available after the assignment on the top level, thanks to the sequence-based two-level storage policy. Recall that A_c is the set of arcs connecting two contiguous storage locations and each of them may contribute to the available storage capacity, via extra storage on their top level, whenever both i and j are unassigned storage locations, as guaranteed by constraints (3) and (4). Here b^{max}_{ij} does denote the maximum among the b^k_{ij} availabilities for the arc (i,j). The third sum in (1) involves the control parameter w, which is set in such a way to weight the relative relevance of the secondary optimization goal (i.e., the assignment of the storage locations belonging to the department sets $\mathcal{D}^s_1, \ldots, \mathcal{D}^s_n$ to the special product types in $\mathcal{K}^s_1, \ldots, \mathcal{K}^s_n$, respectively) with respect to the primary optimization goal. In this sum, the parameter δ^k_i is 1 if the storage location i belongs to a preferable department for the special product type k, and it is 0 otherwise.

Constraints (2) state that each available storage location will be either assigned to a product type or will remain available after the assignment. Constraints (5) define a directed path from Σ to Θ for each $k \in \mathcal{K}$ by means of the binary variables x^k_{ij}, with the aim of modeling the assignment of a sequence of storage locations to each k. In a standard way, each path is modeled in terms of a unitary flow sent from the source node Σ to the destination node Θ. $N^+(i)$ and $N^-(i)$ denote the sets of nodes linked to i via an exiting and an entering arc, respectively, that is $N^+(i) = \{j \in N:\ \exists\ (i,j) \in A\}$ and $N^-(i) = \{j \in N:\ \exists\ (j,i) \in A\}$. Constraints (6) state that a node i can belong to the path designed for the product type k if

and only if the corresponding storage location has been assigned to k. Constraints (7)-(8) are the flow conservation constraints, related to the flow variables f_{ij}^k, which are used to model the storing operations along the sequence of storage locations assigned to each $k \in \mathcal{K}$. Notice that (8) also model the sequence-based two-level storage policy via the second addendum of the right hand side. In fact, if j is the predecessor of i along the path, and it is physically contiguous to i, i.e., $(j,i) \in A_c$, then an extra storage b_{ji}^k can be exploited on the upper level, above j and i, depending on k. Otherwise, only the capacity c_i^k of the storage location i can be exploited. Also observe that, in addition to model the storing operations along the sequence of storage locations for each k, the flow variables and the related constraints prevent subtours in the returned solutions. Constraints (9)-(10) link together sequencing and flow variables. Specifically, a flow of type k can be sent along an arc (i,j) only if (i,j) belongs to the path assigned to k. In that case, if $(i,j) \in A$ with $j \neq \Theta$, i.e., f_{ij}^k represents the amount of product k still to be stored after the storing operations at i, then such an amount must be nonnegative and less than or equal to the total amount q^k to be stored. As for the arcs of type (i,Θ), since they model the end of the storing operations, their flow must be nonpositive and bounded from below by $-2c^{max}$, where c^{max} denotes the maximum capacity of all storage locations in the warehouse. Notice that, for each k, the opposite of the flow along (i,Θ), i.e., $-f_{i\Theta}^k$, indicates the number of items of product type k which can still be stored at i and, possibly, on top of i and its predecessor along the sequence, thus providing an useful additional information about the status of the warehouse in terms of storage availability.

The sets, the parameters and the variables related to the model are summarized in Table 1.

4.5 Valid inequalities

In order to enhance the multicommodity flow model (1)-(10), we have considered and experimented the valid inequalities stated in Lemma 4.1.

Lemma 4.1 *The following equalities and inequalities:*

$$z_i^k = \sum_{j \in N^+(i)} x_{ij}^k \qquad \forall\ k \in \mathcal{K},\ \forall\ i \in \mathcal{S}, \tag{11}$$

$$\sum_{i \in \mathcal{S}} c_i^k z_i^k + \sum_{(i,j) \in A_c} b_{ij}^k x_{ij}^k \geq q^k \qquad \forall\ k \in \mathcal{K}, \tag{12}$$

$$x_{ij}^k \leq z_i^k \qquad \forall\ k \in \mathcal{K},\ \forall\ (i,j) \in A_c, \tag{13}$$

$$x_{ij}^k \leq z_j^k \qquad \forall\ k \in \mathcal{K},\ \forall\ (i,j) \in A_c, \tag{14}$$

are satisfied by any feasible solution of the multicommodity flow model (1)-(10).

Proof: Equalities (11) can be immediately obtained by combining constraints (5) and (6).

Table 1: Sets, parameters and variables used in the model.

	Sets
$\mathcal{K}$	set of different product types
$\mathcal{K}_1^s, \ldots, \mathcal{K}_n^s$	subsets of special product types
$\mathcal{D}_1^s, \ldots, \mathcal{D}_n^s$	subsets of departments where special product types have to be preferably stored
$\mathcal{S}$	set of available storage locations
A_c	set of arcs connecting pairs of contiguous storage locations
	Parameters
Σ	fictitious source node
Θ	fictitious sink node
c_i^k	capacity of storage location i for product type k
b_{ij}^k	capacity made available on top of i and j for product type k, if $(i,j) \in A_c$
δ_i^k	indicate if storage location i is preferable for the special product type k
w	relative relevance of the secondary optimization goal w.r.t. the primary goal
q^k	number of items to stock for product $k \in \mathcal{K}$
c_i^{max}	maximum capacity of storage location i
b_{ij}^{max}	maximum availability of arc (i,j)
c^{max}	maximum capacity of all storage locations in the warehouse
	Variables
$z_i^k \in \{0,1\}$	model the assignment of storage location i to product type k
$z_i^a \in \{0,1\}$	model the availability of storage location i after the assignment
$z_{ij}^a \in \{0,1\}$	model the extra storage on top of i and j available after the assignment
$x_{ij}^k \in \{0,1\}$	model the directed path from node Σ to node Θ for product type k
$f_{ij}^k \in \mathbb{R}$	model the storing operations along the sequence assigned to k in terms of a flow

Regarding (12), let us fix $k \in \mathcal{K}$. From constraints (6) and (8) it follows that

$$\sum_{j \in N^-(i)} f_{ji}^k - \sum_{j \in N^+(i)} f_{ij}^k = c_i^k z_i^k + \sum_{\substack{j \in N^-(i): \\ (j,i) \in A_c}} b_{ji}^k x_{ji}^k$$

holds for any $i \in \mathcal{S}$. Then, by summing up these equalities over i we get

$$\sum_{j \in N^+(\Sigma)} f_{\Sigma j}^k - \sum_{i \in N^-(\Theta)} f_{i\Theta}^k = \sum_{i \in \mathcal{S}} c_i^k z_i^k + \sum_{(i,j) \in A_c} b_{ij}^k x_{ij}^k.$$

Moreover, constraints (7) and (10) guarantee that

$$\sum_{j \in N^+(\Sigma)} f_{\Sigma j}^k - \sum_{i \in N^-(\Theta)} f_{i\Theta}^k \geq \sum_{j \in N^+(\Sigma)} f_{\Sigma j}^k = q^k,$$

thus implying inequalities (12).

Finally, let $k \in \mathcal{K}$ and $(i,j) \in A_c$. Constraints (6) imply

$$z_j^k = \sum_{p \in N^-(j)} x_{pj}^k \geq x_{ij}^k,$$

while equalities (11) imply

$$z_i^k = \sum_{p \in N^+(i)} x_{ip}^k \geq x_{ij}^k,$$

i.e., inequalities (13) and (14) are satisfied. □

The multicommodity flow formulation has been tested with and without the additional constraints (11)-(14). As reported in Section 6, the proposed valid inequalities allowed one to enhance the formulation regarding the number of instances solved. Hereafter model (1)-(10) and model (1)-(14) will be referred to as S2L and S2L-VI, respectively.

It is possible to observe that, if the auxiliary graph is acyclic or can be converted into an acyclic structure, as in the addressed case study, then the flow variables f_{ij}^k and the related flow constraints (7)-(10) can be dropped by S2L-VI. In fact, constraints (12) ensure sufficient capacity to each product type, also taking into account the two level storage policy, and no cycle elimination is required. The resulting strengthened formulation will be referred to as S2L-Acyclic, to emphasize its suitability for the special case of acyclic structured graphs.

4.6 Relaxations

In order to evaluate the quality of the solutions found by the formulations, two relaxations have been proposed starting from S2L-VI. The first one disregards the sequencing constraints, just assigning a set of storage locations to each product type. Notice that the assignment of special product types to preferable departments and the sequence-based two-level storage policy are still managed, the latter although in a relaxed form. The resulting relaxation, called REL-ASSIGN, can thus be stated as follows:

$$\max \sum_{i \in \mathcal{S}} c_i^{max} z_i^a + \sum_{(i,j) \in A_c} b_{ij}^{max} z_{ij}^a + w \sum_{k \in \bigcup_{p=1}^{n} \mathcal{K}_p^s} \sum_{i \in \mathcal{S}} \delta_i^k z_i^k \tag{1}$$

subject to:

$$z_i^a + \sum_{k \in \mathcal{K}} z_i^k = 1 \qquad \forall\, i \in \mathcal{S}, \tag{2}$$

$$z_{ij}^a \leq z_i^a \qquad \forall\, (i,j) \in A_c, \tag{3}$$

$$z_{ij}^a \leq z_j^a \qquad \forall\, (i,j) \in A_c, \tag{4}$$

$$\sum_{i \in \mathcal{S}} c_i^k z_i^k + \sum_{(i,j) \in A_c} b_{ij}^k x_{ij}^k \geq q^k \qquad \forall\, k \in \mathcal{K}, \tag{12}$$

$$x_{ij}^k \leq z_i^k \qquad \forall\, k \in \mathcal{K},\ \forall\, (i,j) \in A_c, \tag{13}$$

$$x_{ij}^k \leq z_j^k \qquad \forall\ k \in \mathcal{K},\ \forall\ (i,j) \in A_c. \tag{14}$$

The second relaxation, instead, relies on the aggregation of the product types. Specifically, we define a representative for each special product type, i.e., $\mathcal{K}_1^s, \ldots, \mathcal{K}_n^s$, and a representative for the set of the remaining product types, i.e., $\mathcal{K} \setminus \bigcup_{p=1}^{n} \mathcal{K}_p^s$. The demand of each representative is set equal to the total demand of the product types in the corresponding set. Model S2L-VI is then modified accordingly. The resulting relaxation will be called REL-AGGR.

4.7 Computational complexity

We conclude this section with a proof of the time complexity of S2L-SLAP problem.

Theorem 4.2 *S2L-SLAP is NP-Hard.*

Proof: The proof is by reduction from the Maximum Fixed-Length Disjoint Paths Problem (MFLDP), which is NP-complete in its decisional form (Garey and Johnson, 1979). Consider an instance of MFLDP, say P_1. Given a directed graph $G = (V, E)$, vertices s and t, and positive integers $K, Q \leq |V|$, P_1 asks to verify whether G contains K mutually vertex-disjoint directed paths from s to t, each involving exactly Q arcs.

Given P_1, define the following instance of S2L-SLAP, hereafter denoted as P_2. In P_2, there are K product types requiring storage (i.e., $|\mathcal{K}| = K$), none of them being special (i.e., $\mathcal{K}_1^s = \cdots = \mathcal{K}_n^s = \emptyset$). Regarding the available storage locations, in P_2 there is an available storage location for each node $i \in V$, with $i \neq s, t$, referred to as i, too. Moreover, there are $2K$ additional storage locations, named $s^1, \ldots, s^K$ and $t^1, \ldots, t^K$, which correspond to K copies of node s in P_1 and to K copies of node t in P_1, respectively. The capacity of each storage location is equal to 1 independently of the product type (i.e., $c_i^k = 1$ for any storage location i and $k \in \mathcal{K}$). Regarding the set of pairs of storage locations that can be consecutive in a sequence, in P_2 the set $\mathcal{P}$ contains a pair for each $(i, j) \in E$ such that both i and j are different than s and t. In addition, $\mathcal{P}$ contains the pairs of storage locations (s^h, i), with $h = 1, \ldots, K$, for each arc $(s, i) \in E$, and the pairs of storage locations (i, t^h), with $h = 1, \ldots, K$, for each arc $(i, t) \in E$. Parameters b_{ij}^k are set in such a way that they hold M, which is a very big value, for the pairs of storage locations (s^h, i) and (i, t^h), for any $h = 1, \ldots, K$ and $k \in \mathcal{K}$, and are 0 otherwise. Finally, the demand of a product type k is $q^k = 2M + Q + 1$ for any $k \in \mathcal{K}$.

By construction, G contains K mutually vertex-disjoint directed paths from s to t, each involving exactly Q arcs, i.e., the instance P_1 is feasible, if and only if in instance P_2 it is possible to assign a sequence of storage locations to each product type by exactly satisfying its demand. Equivalently, P_1 is feasible if and only if the residual capacity of the warehouse after the assignment is $|V - 2| + 2K - (Q + 1)K$. The thesis follows[1]. □

[1] MFLDP aims at defining K or more disjointed paths. Nevertheless, also the variant of this problem where exactly K disjointed paths are sought is NP-complete.

Notice that the sequence-based two-level storage policy comes into play in the presented proof.

5 Matheuristic approaches

Two matheuristic approaches have been designed starting from S2L-VI. The first approach rapidly constructs a solution of good quality via problem decomposition, and then provides it to model S2L-VI as a starting solution, in order to improve it. The second approach, instead, first assigns storage locations to the incoming items via the relaxation REL-ASSIGN presented in Section 4.6, and then determines a sequencing of the assigned storage locations, solving a restricted version of model S2L-VI itself.

5.1 A decomposition approach

Preliminary experiments using the state-of-the-art commercial solver CPLEX have shown that a huge amount of time was spent sometimes by the solver to find a first feasible solution to start the resolution process. In order to overcome this issue, we propose the following two-phase approach:

Phase 1: generate an initial feasible solution to provide to the solver via problem decomposition;

Phase 2: call CPLEX to solve S2L-VI (see Section 4.5), starting from the solution determined in Phase 1.

Phase 1 consists of the following steps:

1. The set $\mathcal{K}$ of the product types is partitioned into Λ subsets, where Λ is a parameter of the approach, in such a way that the number of items to be stored for each subset is about the same, i.e., approximately $(\sum_{k\in\mathcal{K}} q^k)/\Lambda$. Let $\mathcal{K}_\lambda$, with $\lambda = 1, \ldots, \Lambda$, denote the resulting subsets.

2. Λ subproblems are generated and solved in cascade to stock the subsets of product types in $\mathcal{K}_1, \ldots \mathcal{K}_\Lambda$, respectively, each time removing those storage locations already assigned in the previous solved subproblems. Specifically, by denoting by Φ_λ the set of storage locations assigned to the product types in $\mathcal{K}_\lambda$ when solving the λ-th subproblem, then $\Phi_1 \cup \cdots \cup \Phi_{\lambda-1}$ are removed from the set of storage locations $\mathcal{S}$ when determining Φ_λ. So doing, it is not possible to assign the same storage location in different subproblems. All the generated Λ subproblems are solved via model S2L-VI.

3. The optimal solutions of the Λ subproblems are merged into a unique solution, which is the initial feasible solution provided to the Branch and Bound algorithm of CPLEX to solve the overall S2L-VI formulation.

The matheuristic approach is summarized in Algorithm 1. Hereafter it will be named MATD, where D stands for decomposition.

Algorithm 1 The MATD approach

1: Partition $\mathcal{K}$ into Λ almost balanced groups with respect to the number of items to stock: $\mathcal{K}_1, \ldots, \mathcal{K}_\Lambda$
2: $\Phi_0 = \emptyset$
3: **for** $\lambda = 1, \ldots, \Lambda$ **do**
4: Remove $\Phi_{\lambda-1}$ from $\mathcal{S}$
5: Solve the λ-th subproblem on $\mathcal{K}_\lambda$
6: Insert into Φ_λ the assigned storage locations
7: **end for**
8: Unify the subproblem solutions: $\Phi = \bigcup_{\lambda=1}^{\Lambda} \Phi_\lambda$
9: Solve model S2L-VI starting from solution Φ

5.2 An assignment based approach

Also the second approach is composed of two phases. Firstly, an assignment of storage locations to the incoming items is devised via the relaxation REL-ASSIGN (see Section 4.6). Then, a sequencing of the assigned storage locations is looked for by solving a restricted version of model S2L-VI, where the assignment variables are fixed according to the solution found by REL-ASSIGN. Hereafter this approach will be denoted as EUR-RA.

Algorithm 2 The EUR-RA approach

1: Find an assignment $\Phi^{REL-ASSIGN}$ of storage locations to incoming items via relaxation REL-ASSIGN
2: Solve model S2L-VI by fixing the assignment variables according to $\Phi^{REL-ASSIGN}$

Notice that, whereas MATD returns a feasible solution at the end of the first phase, and this can be relevant in case a problem solution needs to be rapidly designed, the solution found in the first phase of EUR-RA is not feasible to S2L-VI. However, if an optimal solution of REL-ASSIGN is determined in the first phase of EUR-RA, and a feasible solution for S2L-VI is found in the second phase, then the solution returned by EUR-RA is provably optimal to S2L-VI. The inconvenience is that, as reported in Section 6, REL-ASSIGN may be computationally hard to solve to optimality. In our experimentation we thus set a time limit to the REL-ASSIGN resolution. As a consequence, if the optimality of the solution computed

by REL-ASSIGN in the first phase of the approach is not certified, then the solution returned by EUR-RA is not necessarily optimal to S2L-VI. However, as shown in Section 6, EUR-RA proved to be very efficient in determining solutions near to the optimal ones, in some cases certifying their optimality.

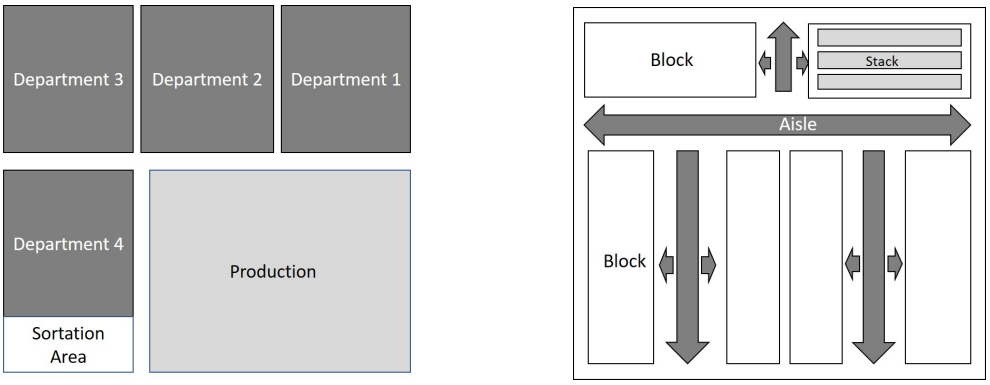
6 Numerical experiments

6.1 The case study addressed

The production site of the considered company, leader in the tissue sector, is composed of a production area, a warehouse, a sortation area and several shipping docks. The warehouse is larger than 10,000 m^2 and comprises four departments. Each department has a rectangular internal layout composed of blocks of storage locations, which are stacks in the considered application. Stacks are accessible only frontally, and they are framed by narrow storage aisles and wide cross aisles. Different blocks may be composed of different number of stacks, all having though the same capacity. However, stacks belonging to different blocks may have different capacities. Specifically, the storage area is divided into 29 blocks, which are composed of a variable number of stacks ranging from 15 to 65. Stacks have a capacity ranging from 8 to 17 items at the ground level, independently on the product type to store.

The sortation area is used as a collection area where items can be temporarily stocked, once retrieved from their positions within the warehouse, waiting to be loaded on the trucks. It can stock up to 1,000 items and is normally filled up as much as possible during the night to quickly start the truck loading operations the next morning. The structure of the warehouse is depicted in Figure 5a, while the internal structure of a department is depicted in Figure 5b.

The production site works daily on 3 shifts of 8 hours. More than 300 different product types can be produced. Two sets of special product types are considered: high rotational and perishable products. High rotational and perishable products have a favorite department.



(a) Warehouse representation (b) Department representation

Figure 5: Warehouse and department representations.

Precisely, and referring to Figure 5a, Department 4, which is a kind of fast picking area since it is located near the sortation area, should be preferable dedicated to high rotational products, while Department 1 should be preferable dedicated to perishable products for its lower tax of humidity. Hereafter high rotational products will be denoted by HR, while perishable products, which are tissues in the considered context, will be denoted by P. Products are released by the production area in small quantities and constantly during each shift, already wrapped and arranged in so-called *columns* of pallets all having the same dimension. That is, items are columns of equal size in the addressed case study, and therefore the inventory will be expressed in terms of columns.

The planning horizon to be considered in this application context is the day. On average, about 900 columns are released per day (300 on average for each shift) and about the same quantity is shipped. Departments and sortation area are thus steadily near to their saturation level. The list of the product types released per day, together with the associated number of columns to store, is known and will be referred to as the *storage list* in the following. As described in Section 3, for each product type included in the daily storage list, a sequence of empty stacks available in the warehouse must be assigned to the product type, which must be suitable to guarantee the storage of all the columns of that product type. Regarding the sequencing decisions, in the considered application any pair of available stacks can be consecutive in a sequence. However, to break symmetries, the set $\mathcal{P}$ contains only arcs of form (i, j) with $i < j$. The auxiliary graph is thus converted into an acyclic structure in the considered case study. Moreover, due to the layout of the warehouse, the set A_c, defining the subset of the arcs connecting contiguous storage locations, contains an arc $(i, j) \in A$ only if both i and j belong to the same block of the warehouse and they are physically contiguous.

Stack assignment and sequencing decisions must respect all the rules presented in Section 3. In particular, whenever two contiguous stacks are occupied by columns of the same product type at the ground level, an additional stack can be created on the top of the two stacks, with the same capacity of each ground stack at the basis. Recall in fact that contiguous stacks belong to the same block of the warehouse and therefore have equal capacity.

6.2 Plan of the experiments

Three types of experiments have been performed. The aim of the first set of experiments is to compare the efficacy and the efficiency of the formulations presented in Sections 4.4 and 4.5. Specifically, S2L and S2L-VI are tested on a wide pool of real instances related to the addressed case study. Moreover, given the acyclic structure characterizing the auxiliary graph in such instances, as outlined in Section 6.1, the strengthened formulation S2L-Acyclic has been tested as well. The set of instances is described in Section 6.3, while the computational results are reported in Section 6.4. The same set of instances has been solved by considering

both the relaxations proposed in Section 4.6 and the continuous relaxation. The obtained results are reported in Section 6.5.

The second set of experiments aims at comparing the performance of the proposed matheuristic approaches MATD and EUR-RA, presented in Sections 5.1 and 5.2, over the same set of instances used in the first set of experiments. The results are reported in Section 6.6.

Finally, in Section 6.7 we assess the efficacy and the efficiency of EUR-RA, which has shown a very good behavior in the second set of experiments, in a wider and more complete setting, where assignment and sequencing storage location problems are consecutively solved, on a daily basis, jointly with picking and housekeeping operations, by thus simulating the use of this matheuristic approach on a 6-day week.

The mathematical formulations and the algorithms have been implemented using the OPL language. The solver used is CPLEX 12.6 (IBM ILOG, 2016). All the experiments have been conducted on an Intel Xeon 5120 computer with 2.20 GHz and 32 GB of RAM, with a time limit of one hour for the experiments related to S2L, S2L-VI, S2L-Acyclic, MATD and EUR-RA, and a time limit of two hours for the relaxation bound computation.

6.3 The instances

The data set provided by the company comprises the following daily information: the warehouse configuration at the beginning of the considered day (i.e., product types and corresponding number of columns inside the warehouse), the storage list of the day (i.e., product types and corresponding number of columns needing storage that day) and the shipping list of the day (i.e., product types and number of columns leaving the warehouse that day).

For the first and second set of experiments, we randomly selected 20 not consecutive days from the data set, and partitioned them into two classes depending on the total amount of columns to stock. The first class, called ClassLA, refers to 10 days where the number of columns to stock is lower than the average number of columns to stock over the 20 selected days. The second class, called ClassHA, refers to 10 days where the number of columns to stock is higher than the average. More in detail, the instances in ClassLA have to assign 737 columns of 11 different product types on average: 39.7% are columns of type HR, whereas 1.6% are columns of type P. The instances in ClassHA have to assign 1098 columns of 14 different product types on average: 20.9% are columns of type HR, whereas 1.3% are columns of type P. The features of the 20 instances are shown in Table 2, where the total number of columns to assign and the number of columns of product type HR and P are reported. Averages are put in the last line of the table.

The average number of empty stacks before the assignment is similar in the two classes: 147 in ClassLA and 139 in ClassHA. Therefore, the size of the auxiliary graph used to model the problem is about the same on average for the two classes of instances. However, the instances in ClassHA have more columns to assign, which are related to a greater number

Table 2: Features of the instance set.

ClassLA				ClassHA			
ID	Total (col.)	HR (col.)	P (col.)	ID	Total (col.)	HR (col.)	P (col.)
1	827	477	0	11	1062	328	16
2	872	364	17	12	1253	462	7
3	575	197	5	13	1050	588	6
4	781	640	6	14	966	419	13
5	973	199	14	15	1151	250	20
6	675	347	16	16	974	12	29
7	457	142	12	17	1019	12	10
8	656	351	21	18	1187	93	23
9	773	81	24	19	1125	0	19
10	784	129	5	20	1192	134	10
Avg.	737	293	12	Avg.	1098	230	15

of different product types. This implies more commodities to manage on average in the corresponding MILP formulations, and so more paths to design. These characteristics render the stack assignment and sequencing particularly heavy for ClassHA, as reported next. As an additional information, the average number of variables over the 20 instances is 271,612, while the average number of constraints is 411,742 for formulation S2L and 416,622 for formulation S2L-VI. Instead, the average number of variables is 136,802, while the average number of constraints is 10,465 for formulation S2L-Acyclic.

6.4 Efficacy and efficiency of S2L, S2L-VI and S2L-Acyclic

Formulations S2L, S2L-VI and S2L-Acyclic rely on the parameter w, which weights the relative relevance of the secondary optimization goal (i.e., the assignment of the preferable departments to the special product types) with respect to the main goal for the company (i.e., to maximize the total space available after the assignment). As outlined in the case study description (Section 6.1), columns of product type HR should be preferably stored in Department 4, whereas columns of product type P should be preferably stored in Department 1. Increasing the value of w would tend to enhance the assignment of these special products to their target departments. We performed several preliminary tests with different values of w, ranging from 1 to 50. Increasing the value of w generally makes the problem harder to solve. This is reasonable since the solver has to maximize the total space available after the assignment while trying to enhance the assignment of columns of types HR and P to their target departments, an objective which may be in contrast with the first one. The two objectives thus become more and more conflicting as w increases. We report here the results related to $w = 5$, which proved to be a good compromise between the total space available af-

Table 3: Comparison of percentage optimality gaps among S2L, S2L-VI and S2L-Acyclic.

ClassLA				ClassHA			
ID	S2L	S2L-VI	S2L-Acyclic	ID	S2L	S2L-VI	S2L-Acyclic
1	2.30%	3.52%	**1.30**%	11	-	-	**2.32**%
2	5.56%	5.66%	**2.48**%	12	6.95%	6.59%	**3.18**%
3	1.12%	1.45%	**0**%	13	3.36%	3.46%	**1.22**%
4	-	3.08%	**1.05**%	14	3.75%	2.54%	**0**%
5	5.28%	4.87%	**1.77**%	15	6.71%	9.88%	**3.80**%
6	1.35%	1.26%	**0**%	16	8.70%	5.61%	**4.21**%
7	1.03%	1.37%	**0.37**%	17	3.88%	6.65%	**2.41**%
8	0.91%	0.84%	**0**%	18	-	-	**2.25**%
9	-	4.24%	**1.79**%	19	7.51%	5.37%	**1.64**%
10	-	3.06%	**1.86**%	20	3.77%	5.13%	**2.47**%

ter the assignment and the number of columns assigned to their target departments, without excessively complicating the resolution process of the instances.

Table 3 shows, for S2L, S2L-VI and S2L-Acyclic and for each of the 20 instances belonging to ClassLA or to CLassHA, the percentage optimality gap returned by the solver at the end of the resolution process, which may occur either because the optimum is found or because the total time limit (i.e., one hour) is reached. Notice that, for S2L-Acyclic, the resolution of four instances stopped due to the first motivation. Sometimes CPLEX was not able to find a feasible solution within the time limit imposed, in which case '-' appears in the table for the corresponding instance.

As reported in the table, the instances in ClassHA appear to be more difficult to address computationally, according to their characteristics outlined in Section 6.3. Regarding the tested formulations, results are pretty similar when considering S2L or S2L-VI on the instances that both formulations are able to solve, sometimes achieving better results for the former (e.g., Instance 1), sometimes being better for the latter (e.g., Instance 19). Nevertheless, within the time limit imposed, CPLEX was not able to find a feasible solution for five instances out of 20 when the formulation S2L is considered, whereas only two instances were not solved when the formulation S2L-VI is taken into account. On the other hand, all the instances were solved when using the strengthened formulation S2L-Acyclic. Indeed, S2L-Acyclic is able to solve four instances to optimality, three belonging to ClassLA and one to ClassHA, and overally returns lower percentage optimality gaps than the ones returned by S2L and S2L-VI. Since S2L-Acyclic achieved better results compared to S2L and S2L-VI, it has been chosen as the kernel formulation to implement the resolution approaches proposed in Sections 5.1 and 5.2.

6.5 Comparison with relaxed problems

We consider three relaxations to assess the difficulties in solving the problem. The first one is REL-AGGR, which relies on the aggregation of the product types into three categories: HR, P and the remaining product types. Notice, however, that the number of columns to assign is the same as in the not-relaxed case. The second relaxation is REL-ASSIGN, which disregards the sequencing constraints, just assigning a set of storage locations to each product type. Both relaxations have been described in Section 4.6. Given the acyclic structure of the auxiliary graphs in the considered case study, we implemented REL-AGGR starting from the strengthened formulation S2L-Acyclic. Finally, the third relaxation is the continuous relaxation of S2L-Acyclic, named REL-C in the following.

The three relaxations have been solved with CPLEX with a time limit of two hours. Table 4 reports, for REL-AGGR, REL-ASSIGN and REL-C, and for each of the 20 instances, the solving time in seconds (time, in the table) required by CPLEX, which may stop either by finding an optimum or by reaching the time limit imposed, the percentage optimality gap (gap%, in the table) when CPLEX stops, and either the optimal objective value or the best upper bound found by CPLEX, if the optimum is not reached (UB, in the table). The last column (BB, in the table) reports the best upper bound found through the three relaxations.

As reported in Table 4, REL-AGGR and REL-C solve all the instances to optimality in a few seconds, the latter being faster and overally providing better bounds. Regarding REL-ASSIGN instead, in the majority of the cases CPLEX reaches the time limit of two hours. However, 7 instances are solved to optimality. Moreover, REL-ASSIGN always returns the best upper bound among the three relaxations (see column BB).

The results related to REL-AGGR suggest that one of the difficulties of the problem addressed is the high number of product types to manage. In fact, all the instances of the relaxed problem modelled by REL-AGGR, where only three product types are present, are solved by CPLEX to optimality, and in a few seconds in the majority of the cases. As opposed, when a realistic number of different product types is considered, as for the cases reported in Section 6.3, the problem becomes more difficult to address computationally. Such an insight has been exploited in designing the matheuristic approach based on decomposition.

On the other hand, the results related to REL-ASSIGN show that when the sequencing aspect is disregarded, then the problem remains generally hard to solve. However, as outlined, 7 instances have been solved to optimality.

6.6 Efficacy and efficiency of matheuristics MATD and EUR-RA

We now compare the results obtained by the matheuristics MATD and EUR-RA on the same set of instances used in the previous experimentation. Given the acyclic structure of the auxiliary graphs, and given the best results achieved (see Section 6.4), the kernel formulation used to implement MATD and the Phase 2 of EUR-RA is S2L-Acyclic.

Table 4: Relaxation bounds.

	REL-AGGR			REL-ASSIGN			REL-C			
ID	time	gap %	UB	time	gap %	UB	time	gap %	UB	BB
1	7.94	0%	1929	2294.14	0%	**1877**	0.45	0%	1927.86	**1877**
2	2.68	0%	1091	3755.02	0%	**975**	0.49	0%	1061.13	**975**
3	1.34	0%	2068	480.20	0%	**1984**	0.41	0%	2041.72	**1984**
4	45.07	0%	3544	7200	0.87%	**3494.00**	3.59	0%	3533.50	**3494.00**
5	26.70	0%	2817	7200	1.11%	**2734.14**	2.26	0%	2776.35	**2734.14**
6	139.53	0%	3734	192.75	0%	**3691**	1.27	0%	3752.62	**3691**
7	33.90	0%	4435	2423.38	0%	**4386**	1.93	0%	4427.10	**4386**
8	5.35	0%	3730	183.35	0%	**3674**	1.83	0%	3709.16	**3674**
9	571.15	0%	3715	7200	2.48%	**3673.81**	3.26	0%	3693.01	**3673.81**
10	53.52	0%	3585	7200	1.07%	**3511.08**	4.90	0%	3544.57	**3511.08**
11	41.17	0%	2639	7200	2.66%	**2587.02**	4.12	0%	2606.69	**2587.02**
12	15.45	0%	2089	7200	2.52%	**1994.00**	1.80	0%	2017.93	**1994.00**
13	14.70	0%	3434	7200	1.47%	**3258.17**	5.85	0%	3298.13	**3258.17**
14	11.98	0%	1464	716	0%	**1368**	0.55	0%	1427.25	**1368**
15	6.78	0%	1486	7200	2.44%	**1376.79**	2.04	0%	1413.31	**1376.79**
16	0.84	0%	1247	7200	2.63%	**1175.10**	0.53	0%	1221.31	**1175.10**
17	2.26	0%	1889	7200	2.06%	**1813.54**	1.28	0%	1847.45	**1813.54**
18	26.35	0%	3431	7200	1.85%	**3334.46**	4.40	0%	3368.33	**3334.46**
19	7.53	0%	2889	7200	1.67%	**2758.28**	2.89	0%	2779.73	**2758.28**
20	37.67	0%	2661	7200	1.27%	**2591.46**	1.55	0%	2630.82	**2591.46**

In the following, we briefly discuss the parameter setting of the approaches, then reporting the results for each approach, firstly in terms of efficiency and then in terms of solution quality.

As described in Section 5.1, MATD consists of two phases: Phase 1, devoted to the construction of an initial feasible solution via problem decomposition, and Phase 2, where S2L-Acyclic is solved starting from the feasible solution determined in Phase 1. We implemented Phase 1 by matching large and small product types to get groups that have approximately the same number of items to store. Moreover, we performed the following tests on the two parameters of the approach, i.e., the number Λ of subproblems in which the original problem is decomposed and the time dedicated to Phase 1 (recall that an overall time limit of 60 minutes has been established by the warehouse managers to solve the problem).

We experimented several values for Λ ranging from 2 to 7 (notice that $\Lambda = 1$ corresponds to the original S2L-Acyclic formulation). $\Lambda = 3$ resulted a good compromise between the time required by CPLEX to solve all the subproblems and the quality of the solutions obtained.

We also experimented two alternative ways to partition the time limit of 60 minutes between the two phases of MATD. We considered a first setting where at most 20 minutes are given to Phase 1 and 40 minutes are given to Phase 2, and a second setting where instead

at most 40 minutes are given to Phase 1 and 20 minutes are given to Phase 2. In both settings, we split the time allocated to Phase 1 among the Λ subproblems proportionally with respect to the number of columns each subproblem has to handle. If a subproblem is solved before reaching the assigned time limit, i.e., an optimum is found for the subproblem, then the remaining amount of time is added to the one allocated to Phase 2. We solved the 20 instances by considering both time allocation settings. The obtained results showed that the subproblems may either be very easy to tackle by CPLEX, in which case a few seconds are needed to find the optimum, or, on the opposite, very tricky to solve. In the latter case, assigning at most 20 minutes to Phase 1 brought some of the subproblems to stop their resolution process too early, with high optimality gaps. This compromised the quality of the solution provided to Phase 2 and, consequently, the quality of the final solution, especially for the instances of ClassHA. As opposed, assigning at most 40 minutes to Phase 1 eliminates this issue for the majority of the subproblems, and has a positive impact in terms of final optimality gap and objective value. Therefore, we report here the results related to $\Lambda = 3$ and to the time setting where 40 minutes are dedicated to Phase 1 and 20 minutes to Phase 2. Moreover, parameter w is set to 5 as discussed in Section 6.4.

As described in Section 5.2, EUR-RA consists of two phases as well: Phase 1, devoted to the definition of a storage location assignment to the incoming items via the relaxation REL-ASSIGN, and Phase 2, where a sequencing of the assigned storage locations is looked for by solving a restricted version of S2L-Acyclic. Several tests have been performed to define the most suitable time setting to Phase 1. Preliminary results showed that devoting at most 20 minutes to Phase 1 leads to good results in terms of optimality gaps, available space and special product types assignment for the considered instances. We thus report here the results for EUR-RA related to the setting where 20 minutes are dedicated to Phase 1. Regarding Phase 2, the best results in terms of computational time have been obtained by stopping the execution as soon as a feasible sequencing is found. Finally, we set $w = 5$ as discussed before.

Table 5 reports the performance of MATD and EUR-RA, separately for the instances in ClassLA and in ClassHA, in terms of solution time (in seconds) and percentage optimality gap, which is calculated by considering the best upper bound reported in Table 4 (column BB). The last row reports the average indicators for the two classes of instances.

As shown in the table, both MATD and EUR-RA achieve very good results in terms of optimality gap for both classes of instances. In particular, both approaches find 7 optimal solutions within the time limit imposed. At this regard notice that, in some cases, the time required by MATD to compute an optimal solution is the maximum allowed: this is due to the fact that the approach is not able to certify the optimality of the computed solution within that time. That is, its best upper bound when the algorithm is stopped is greater than the best upper bound in column BB of Table 4, i.e., the one used to calculate the optimality gap.

Table 5: Performance of MATD and EUR-RA.

ClassLA					ClassHA				
	MATD		EUR-RA			MATD		EUR-RA	
ID	time	gap%	time	gap%	ID	time	gap%	time	gap%
1	3600	0.00%	1200	0.00%	11	3600	2.86%	1214	**2.66**%
2	3600	0.00%	1201	0.00%	12	3600	2.57%	1202	2.57%
3	1636	0.00%	395	0.00%	13	3600	1.47%	1205	1.47%
4	3600	1.01%	1203	**0.87**%	14	3533	0.00%	571	0.00%
5	3600	**0.97**%	1203	1.11%	15	3600	2.82%	1206	**2.44**%
6	3600	0.00%	139	0.00%	16	3600	3.08%	1203	3.08%
7	3600	0.00%	1202	0.00%	17	3600	**2.17**%	1202	2.63%
8	3330	0.00%	172	0.00%	18	3600	2.38%	1205	**2.07**%
9	3600	**2.14**%	1205	2.62%	19	3600	1.89%	1204	**1.67**%
10	3600	**1.18**%	1204	1.24%	20	3600	1.55%	1203	**1.31**%
Avg.	3377	0.53%	912	0.58%	Avg.	3593	2.08%	1141	1.99%

More in detail, both approaches have pretty similar results in terms of average optimality gap in solving the instances of ClassLA, sometimes MATD achieving a lower value (for Instances 5, 9 and 10), in one case instead being EUR-RA better (for Instance 4). Nevertheless, EUR-RA is able to solve all the 10 instances in less time than MATD. Specifically, EUR-RA requires on average about one third of the time needed by MATD. Regarding ClassHA, both the optimality gap and the solution time increase on average for both MATD and EUR-RA with respect to ClassLA. Additionally, only one instance is solved to optimality by both approaches. On this class of instances, however, EUR-RA is generally more effective than MATD, with a lower optimality gap for a half of the instances, and a computational time which is again about one third, on average, than the one needed by MATD.

As remarked in Section 5.2, MATD returns a feasible solution also at the end of Phase 1, feature which can be relevant in case the problem needs to be rapidly solved. As an additional information, the average solving time required by the Phase 1 of MATD is 410.07 seconds for ClassLA and 837.34 seconds for ClassHA, with an average optimality gap of 1.00% and 2.91%, respectively (again, the optimality gaps are calculated by considering the best upper bounds reported in Table 4). Only two optima are found at the end of Phase 1. It is thus evident the impact of Phase 2 to reduce the average optimality gaps obtained at the end of Phase 1, and to increase the number of the optimal solutions found.

Regarding EUR-RA, we report that, for all the instances, Phase 2 was able to compute a solution to S2L-Acyclic starting from the assignment devised in Phase 1. The average time to find a feasible sequencing in Phase 2 is just 3 seconds, no matter if the instances belong to ClassLA or to ClassHA.

Table 6: Quality of the solutions of MATD and EUR-RA.

	ClassLA							ClassHA						
	MATD			EUR-RA				MATD			EUR-RA			
ID	Av. space	HR in Dep.4	P in Dep.1	Av. space	HR in Dep.4	P in Dep.1	ID	Av. space	HR in Dep.4	P in Dep.1	Av. space	HR in Dep.4	P in Dep.1
1	1852	27%	-	1852	27%	-	11	**2415**	96%	100%	2410	96%	100%
2	895	97%	100%	895	97%	100%	12	1889	**54**%	100%	1889	50%	100%
3	1954	77%	100%	1954	77%	100%	13	3106	94%	100%	3106	94%	100%
4	3384	63%	100%	**3389**	63%	100%	14	1308	78%	0%	1308	78%	0%
5	**2658**	**100**%	100%	2654	96%	100%	15	1249	100%	100%	**1254**	100%	100%
6	3611	100%	100%	3611	100%	100%	16	1125	0%	100%	1125	0%	100%
7	4341	83%	100%	4341	83%	100%	17	**1770**	0%	100%	1762	0%	100%
8	3589	100%	100%	3589	100%	100%	18	3217	88%	100%	**3227**	88%	100%
9	**3577**	58%	33%	3550	58%	**100**%	19	2692	-	100%	**2698**	-	100%
10	**3445**	46%	100%	3443	**50**%	100%	20	2512	**67**%	100%	**2528**	37%	100%
Avg.	2931	75%	93%	2928	75%	100%	Avg.	2128	64%	90%	2131	60%	90%

Table 6 reports, for MATD and EUR-RA, and for each of the 20 instances, qualitative indications on the solutions found in terms of space available after the assignment and number of columns of product types HR and P assigned to their target departments at the end of Phase 2. The quality of the solutions computed by MATD and EUR-RA is quite similar. Regarding the secondary objective, the percentage of columns of product types HR and P assigned to their target departments is very high for both the approaches. In particular, both MATD and EUR-RA are able to assign all the columns of product type P to Department 1 in 17 out of 19 instances (notice that there are no columns of product type P to manage in Instance 1, see Table 2, and this is indicated by the symbol '-' in Table 6; similarly, '-' in the row of Instance 19 is due to the fact that there are no columns of product type HR to manage for that instance). By considering MATD, we report that the improvement of such quality indicators from Phase 1 to Phase 2 is 0.33% (Av. space), 4.27% (HR in Dep.4) and 0% (P in Dep.1) for the instances of ClassLA, and 0.57% (Av. space), 1.26% (HR in Dep.4) and 7.14% (P in Dep.1) for the instances of ClassHA. This testifies the already very good quality results obtained by MATD at the end of Phase 1, especially for ClassLA.

We finally report that, although S2L-Acyclic appears to be a very efficient formulation to address S2L-SLAP when the auxiliary graph has an acyclic structure, the proposed matheuristics allow one to improve, on average, all the indicators discussed above. In particular, the average percentage gap over the 20 instances is 1.49% for S2L-Acyclic, while it is 1.30% for MATD and 1.21% for EUR-RA. Notice that, to allow the comparison, the gaps for S2L-Acyclic have been computed here by considering the best upper bounds in Table 4, as for MATD and EUR-RA. Regarding the primary goal of the company, the improvement achieved by MATD in terms of available space after the assignment is +0.14% for ClassLA and +2.84%

for ClassHA, while it is +2.30% for ClassLA and +3.03% for ClassHA by considering EUR-RA. Finally, regarding the secondary goal of the company, indeed S2L-Acyclic often equals the results obtained with the matheuristics concerning the number of columns of special product types assigned to their target departments, but generally MATD and EUR-RA return slight better results, especially for the hardest instances and considering the number of columns of product type HR assigned to Department 4. In particular, for ClassHA, MATD always provides the best result in terms of this indicator.

Our conclusion is that both matheuristics appear to be valuable tools to tackle problem S2L-SLAP within the time limit imposed.

By summarizing, the reported results suggest the following observations:

- ClassHA appears to be harder to address than ClassLA for all the approaches;
- The strengthened formulation S2L-Acyclic seems to be well able to exploit the graph structure of the case study addressed, and it appears to be preferable to S2L and to S2L-VI; in particular, it solves all the instances within the stated time limit, four of them to optimality, and overally it returns lower percentage optimality gaps;
- S2L-Acyclic shows to be efficient, and it returns solutions of good quality in terms of both the primary and the secondary goal of the company;
- Both matheuristics improve on average S2L-Acyclic regarding the percentage optimality gap, as well as the available space after the assignment and the percentage of special product types assigned to their target departments;
- Both matheuristics achieve very good results in terms of percentage optimality gap, and they return solutions of similar quality regarding the primary and the secondary goal for the company;
- EUR-RA is preferable to MATD in terms of solution time, being indeed very fast;
- MATD is able to construct a feasible solution of good quality during its Phase 1, in less than 15 minutes, and sometimes it appears to be preferable in terms of the assignment of the product type HR to its target department.

6.7 Simulating stack assignment and sequencing over a week

In order to assess the impact of the proposed methodology in a real logistic environment, we have selected EUR-RA, which has shown very good results in the performed tests, and have analyzed its behavior by simulating the typical warehouse operations in a weekly time horizon. We simulated the logistic process as shown in Algorithm 3.

We start with a realistic configuration of the warehouse at Day 1 (i.e., the first day of the considered week). The initial positions of the columns in the warehouse are randomly generated by respecting some agreed industrial practice or insights given by the company. At the beginning of each Day i, columns of product types specified in the shipping list are picked one stack at a time, according to the FIFO retrieving order, and moved to the sortation area until its saturation (see Line 2). The set $\mathcal{S}$ of the empty stacks is thus defined (Line

Algorithm 3 One week simulation

Input data: Initial configuration for Day 1, shipping and storage lists for Days 1 to 6

1: **for** Day $i = 1, \ldots, 6$ **do**
2: Fill the sortation area till saturation with products in the shipping list of Day i
3: Define the set of available storage locations $\mathcal{S}$
4: Solve the assignment and sequencing problem for Day i with Algorithm 2
5: Fill the warehouse according to the algorithm solution
6: Perform some housekeeping operations
7: **end for**

3). We solve the S2L-SLAP related to Day i by means of EUR-RA (Line 4). Then, the warehouse is updated by filling the assigned stacks in accordance with the solution found by EUR-RA (Line 5). At the end of the day, some housekeeping operations are performed (Line 6), to enhance the space available in the warehouse before the next assignment. Specifically, in order to exploit as much as possible the sequence-based two-level storage policy for next assignments, the housekeeping operations consist in: (i) making empty stacks contiguous within each block as much as possible; (ii) moving groups of occupied stacks from one block to another one whenever possible (provided that stacks have the same capacity). The final configuration obtained for Day i then becomes the initial configuration for Day $i + 1$.

According to the results in Section 6.6, we used EUR-RA by allocating 20 minutes to Phase 1, stopping Phase 2 as soon as a feasible sequencing is found, and setting $w = 5$. Table 7 reports information on the storing and shipping lists for the considered week, composed of three days in ClassLA and three days in ClassHA (shipping and storage operations are not planned on Sunday).

We summarize in Table 8 the results related to the performance of EUR-RA and the quality of the solutions obtained in terms of primary and secondary goals. Specifically, for each day of the week we report the running time in seconds (second column), and the optimality gap (third column), calculated by considering the best upper bound obtained from REL-

Table 7: Storing and shipping lists for the selected week.

	Storage list			Shipping list		
Day	Total (col.)	HR (col.)	P (col.)	Total (col.)	HR (col.)	P (col.)
1	856	328	16	1078	541	8
2	1120	359	20	1282	556	20
3	1069	533	25	1212	624	22
4	1012	737	59	1094	497	21
5	811	670	6	947	521	15
6	156	103	0	0	0	0

Table 8: Simulation on a week: performance of EUR-RA algorithm.

			Before assignment		After assignment		After housekeeping			
Day	time	gap %	Occup.	Avail	Occup.	Avail	Occup.	Avail	HR in Dep.4	P in Dep.1
1	20	0%	16209	2789	17065	1887	15987	2270	96%	100%
2	1207	2.09%	15987	2815	17107	1620	15825	2261	85%	100%
3	114	0%	15825	2872	16894	1778	15682	2269	95%	100%
4	1209	2.64%	15682	2946	16694	1782	15600	2227	91%	84%
5	80	0%	15600	2328	16411	1513	15464	1863	82%	100%
6	0.29	0%	15464	1863	15620	1676	15620	1676	100%	-

ASSIGN within two hours. Then, we report (in columns): the number of occupied and available columns immediately before the storage location assignment (i.e., after Step 2 in Algorithm 3); the number of occupied and available columns immediately after the storage location assignment (i.e., after Step 4 in Algorithm 3); and the number of occupied and available columns after the housekeeping operations (i.e., after Step 6 in Algorithm 3). The number of the available columns, i.e., the indicator Avail., has been computed according to the specification of the objective function of the proposed mathematical formulation, i.e., (1). Finally (last two columns), we report the daily percentages of columns of product type HR assigned to Department 4 and those of product type P assigned to Department 1.

EUR-RA seems to be successful in assigning and sequencing stacks also on a weekly basis. It determines an optimal solution for four days out of six, and in the remaining two days it achieves very good results in terms of optimality gap (being lower than 3%). Moreover, its total solution time is small, much lower than the one hour time limit per day imposed.

Since at most 21,299 columns can be stocked in case a unique product type would be present, in the considered week departments and sortation area are daily near to their saturation level. This testifies the efficacy of the proposed matheuristic in computing solutions of high quality regarding the space availability. By considering the secondary optimization goal, a very high percentage of columns of product type HR is assigned daily to Department 4, while all the columns of product type P are assigned to Department 1 in four out of five days (no columns of type P have to be stored in Day 6). EUR-RA thus appears to be a valuable tool to solve S2L-SLAP in a real application context, also on a weekly time horizon.

7 Conclusions

We address a problem which combines storage location assignment with sequencing decisions about the assigned storage locations. We prove that the problem is NP-Hard, and we model it as a constrained multicommodity flow problem on an auxiliary graph. We then propose a

MILP model, with some valid inequalities, based on the suggested formulation. Two relaxations are presented to estimate the quality of the model solutions. Since the problem can be very hard to address computationally, two matheuristic approaches are then designed starting from the proposed MILP model. A case study is then presented, which is related to the tissue logistics sector and which motivated our research in this topic. Computational experiments on a wide real test bed show the efficiency and the efficacy of the proposed approaches.

We plan to extend the achieved results by studying an optimization problem which integrates the above described assignment and sequencing storage location decisions with a so called pick up and put away problem where, given a fleet of vehicles, decisions related to how and when to move incoming items to their assigned storage locations (put away) and to collect outgoing items for shipping (pick up) need to be made. The assignment and sequencing of storage locations, the scheduling of put away and pick up operations, and the routing of the vehicles inside the warehouse define hard interdependent decisions which are very challenging to address.

Acknowledgements

The authors thank the three anonymous referees and the associate editor for the interesting comments and suggestions, which allowed to greatly improve the previous versions of the paper. The research has been supported by Region of Tuscany-Regional Government (POR FESR 2014-2020-Line 1-Research and Development Strategic Projects) through the Project IREAD4.0 under Grant CUP 7165.24052017.112000028.

References

Accorsi, R., Baruffaldi, G., and Manzini, R. (2017). Design and manage deep lane storage system layout. an iterative decision-support model. *The International Journal of Advanced Manufacturing Technology*, 92(1-4):57–67.

Ang, M., Lim, Y. F., and Sim, M. (2012). Robust storage assignment in unit-load warehouses. *Management Science*, 58(11):2114–2130.

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

Battini, D., Glock, C., Grosse, E., Persona, A., and Sgarbossa, F. (2016). Human energy expenditure in order picking storage assignment: A bi-objective method. *Computers & Industrial Engineering*, 94:147–157.

Bianchi-Aguiar, T., Silva, E., Guimarães, L., Carravilla, M. A., and Oliveira, J. F. (2018). Allocating products on shelves under merchandising rules: Multi-level product families with display directions. *Omega*, 76:47–62.

Bortolini, M., Botti, L., Cascini, A., Gamberi, M., Mora, C., and Pilati, F. (2015). Unit-load storage assignment strategy for warehouses in seismic areas. *Computers & Industrial Engineering*, 87:481–490.

Boysen, N., Boywitz, D., and Weidinger, F. (2018). Deep-lane storage of time-critical items: one-sided versus two-sided access. *OR Spectrum*, 40(4):1141–1170.

Boywitz, D. and Boysen, N. (2018). Robust storage assignment in stack-and queue-based storage systems. *Computers & Operations Research*, 100:189–200.

Carlo, H. J., Vis, I. F., and Roodbergen, K. J. (2014). Storage yard operations in container terminals: Literature overview, trends, and research directions. *European Journal of Operational Research*, 235(2):412–430.

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

Dekker, R., Voogd, P., and Van Asperen, E. (2007). Advanced methods for container stacking. In *Container terminals and cargo systems*, pages 131–154. Springer.

Ene, S., Küçükoğlu, İ., Aksoy, A., and Öztürk, N. (2016). A genetic algorithm for minimizing energy consumption in warehouses. *Energy*, 114:973–980.

Foroughi, A., Boysen, N., Emde, S., and Schneider, M. (2020). High-density storage with mobile racks: Picker routing and product location. *Journal of the Operational Research Society*, pages 1–19.

Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman, San Francisco, CA.

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

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

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

Hübner, A. and Schaal, K. (2017). A shelf-space optimization model when demand is stochastic and space-elastic. *Omega*, 68:139–154.

Larco, J., De Koster, R., Roodbergen, K., and Dul, J. (2017). Managing warehouse efficiency and worker discomfort through enhanced storage assignment decisions. *International Journal of Production Research*, 55(21):6407–6422.

Lehnfeld, J. and Knust, S. (2014). Loading, unloading and premarshalling of stacks in storage areas: Survey and classification. *European Journal of Operational Research*, 239(2):297–312.

Li, J., Moghaddam, M., and Nof, S. (2016). Dynamic storage assignment with product affinity and abc classification—a case study. *The International Journal of Advanced Manufacturing Technology*, 84(9-12):2179–2194.

Maniezzo, V., Boschetti, M. A., and Gutjahr, W. J. (2021). Stochastic premarshalling of block stacking warehouses. *Omega*, 102:102336.

Maschietto, G. N., Ouazene, Y., Ravetti, M. G., de Souza, M. C., and Yalaoui, F. (2017). Crane scheduling problem with non-interference constraints in a steel coil distribution centre. *International Journal of Production Research*, 55(6):1607–1622.

Meneghetti, A. and Monti, L. (2014). Multiple-weight unit load storage assignment strategies for energy-efficient automated warehouses. *International Journal of Logistics Research and Applications*, 17(4):304–322.

Ostermeier, M., Düsterhöft, T., and Hübner, A. (2021). A model and solution approach for store-wide shelf space allocation. *Omega*, 102:102425.

Pang, K.-W. and Chan, H.-L. (2017). Data mining-based algorithm for storage location assignment in a randomised warehouse. *International Journal of Production Research*, 55(14):4035–4052.

Quintanilla, S., Pérez, Á., Ballestín, F., and Lino, P. (2015). Heuristic algorithms for a storage location assignment problem in a chaotic warehouse. *Engineering Optimization*, 47(10):1405–1422.

Ramtin, F. and Pazour, J. (2015). Product allocation problem for an AS/RS with multiple in-the-aisle pick positions. *IIE Transactions*, 47(12):1379–1396.

Revillot-Narváez, D., Pérez-Galarce, F., and Álvarez-Miranda, E. (2020). Optimising the storage assignment and order-picking for the compact drive-in storage system. *International Journal of Production Research*, 58(22):6949–6969.

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

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

Stahlbock, R. and Voß, S. (2008). Operations research at container terminals: a literature update. *OR Spectrum*, 30(1):1–52.

Tang, L., Liu, J., Yang, F., Li, F., and Li, K. (2015). Modeling and solution for the ship stowage planning problem of coils in the steel industry. *Naval Research Logistics*, 62(7):564–581.

Tang, L., Xie, X., and Liu, J. (2014). Crane scheduling in a warehouse storing steel coils. *IIE Transactions*, 46(3):267–282.

Tang, L., Zhao, R., and Liu, J. (2012). Models and algorithms for shuffling problems in steel plants. *Naval Research Logistics*, 59(7):502–524 .

Trindade, M. A., Sousa, P. S., and Moreira, M. R. (2021). Ramping up a heuristic procedure for storage location assignment problem with precedence constraints. *Flexible Services and Manufacturing Journal*, https://doi.org/10.1007/s10696-021-09423-w.

Vis, I. F. and De Koster, R. (2003). Transshipment of containers at a container terminal: An overview. *European Journal of Operational Research*, 147(1):1–16.

Xie, X., Zheng, Y., and Li, Y. (2014). Multi-crane scheduling in steel coil warehouse. *Expert Systems with Applications*, 41(6):2874–2885.

Yang, P., Miao, L., Xue, Z., and Qin, L. (2015). An integrated optimization of location assignment and storage/retrieval scheduling in multi-shuttle automated storage/retrieval systems. *Journal of Intelligent Manufacturing*, 26(6):1145–1159.

Zaerpour, N., Yu, Y., and de Koster, R. B. (2015). Storing fresh produce for fast retrieval in an automated compact cross-dock system. *Production and Operations Management*, 24(8):1266–1284.

Zäpfel, G. and Wasner, M. (2006). Warehouse sequencing in the steel supply chain as a generalized job shop model. *International Journal of Production Economics*, 104(2):482–501.

Zhang, G., Nishi, T., Turner, S., Oga, K., and Li, X. (2017). An integrated strategy for a production planning and warehouse layout problem: Modeling and solution approaches. *Omega*, 68:85–94.