Iterated Greedy Local Search for the Order Picking Problem Considering Storage Location and Order Batching Decisions

Diana L. Huerta-Muñoz[a*]. Roger Z. Ríos-Mercado[b], and Jesús F. López-Pérez [c,d]

[a]Graduate Program in Systems Engineering, Department of Mechanical and Electrical Engineering, Universidad Autónoma de Nuevo León, Mexico;
[b]Graduate Program in Electrical Engineering, Department of Mechanical and Electrical Engineering, Universidad Autónoma de Nuevo León, Mexico;
[c]Department of Business Management and Public Accounting, Universidad Autónoma de Nuevo León, Mexico;
[d]College of Business, Texas A&M International University, USA.

COMPILED
December 11st, 2023

ABSTRACT
Order picking is the process of collecting products from a specific location to complete customer orders. It is the most costly activity inside a warehouse with up to 65% of the incurred costs. Two closely related problems to the order picking are the storage location of products and the order batching, which may affect the routes performed by pickers. In the first one, the decision is where to place the items arriving at the warehouse in several locations and, in the second one, the decision is to group customer orders into batches, which are assigned later to pickers. Although these subproblems are commonly studied and solved independently, their integration may result in greater improvements. We study an integrated picking problem that considers these three subproblems simultaneously. The problem is motivated by a real-world application. The aim is to obtain the best storage location of products, order batching, and picking sequence that minimizes the total picking routing cost. We present a mathematical model and propose an iterated greedy local search metaheuristic. The proposed method is assessed on pseudo-real and real-world instances. Results indicate that instances with up to 8197 order lines, 654 customer orders, and 4252 products, are solved efficiently by the proposed solution method.

Keywords: Order picking; Storage location assignment; Order batching; Picking routing; Integer programming; Metaheuristics.

*D. L. Huerta-Muñoz (corresponding author)**, email: dianahuerta@yalma.fime.uanl.mx, ORCID: 0000-0002-4818-9311. **R. Z. Ríos-Mercado**, ORCID 0000-0003-1053-5183. **J. F. López-Pérez**, ORCID 0000-0002-8283-6359

1. Introduction

In a supply chain, an important element to consider is the distribution center (DC). A DC can be seen as a specific type of warehouse, where several associated activities that take place in this area, have an impact on on-time deliveries to the end customers. Activities such as receiving products, allocating items at a corresponding storage location, grouping several orders into batches, and collecting products to complete customer orders, which are then packaged and delivered to the final customers, are some of the most important decisions for a successful DC performance. If all or some of them are not properly connected, it can generate costly logistic operations and lead to a poor customer perception. Among all these activities, there is one that incurs most of the warehouse costs, called the *Order Picking* (OP).

Order picking is the process of retrieving products from a specific location within a DC in order to complete customer orders. It is the most costly activity inside of a warehouse with up to 65% of the incurred costs (de Koster et al. 2007; Theys et al. 2010). Although all the activities performed within a DC have an impact on the order picking efficiency, there are two activities that are closely related and whose effect is direct and significant, these are *the storage location assignment of products* and *the order batching*. Both activities may affect the *sequence of routes* performed by pickers if they are not well-optimized, or if they are independently optimized, which is usually the case. In the storage location assignment, the decision to consider is where to place the items arriving at the warehouse considering that there are several available locations or racks. In the order batching, the decision is to determine how to group customer orders, which must be assigned later to pickers to perform the corresponding routes. Although these three decisions (storage, batching, and picking) are commonly solved independently, recent studies have analyzed that their integration may result in a greater improvement (Silva et al. 2020).

Focus of the paper. In this work, we aim to study the integration of the storage location of products, order batching, and picker routing decisions, that minimizes the total picking cost within a DC. The study is motivated by a real-world application of a local retail firm, which has several distribution centers that perform a manual *picker-to-parts* order picking, and where changes in the storage location of products can be performed each day.

Contribution. The contribution of this paper is twofold. First, we introduce a mixed-integer linear programming (MILP) model that integrates the storage location of products, order batching, and picker routing decisions. In addition, an affinity criterion is introduced to consider the similarity of the items to be assigned to the storage locations. This provides a valuable dimension to the model and enhances the accuracy of order fulfillment, minimizing picking errors. Second, we develop an Iterated Greedy Local Search (IGLS) metaheuristic to efficiently obtain good-quality solutions for large-scale instances for the integrated problem. The metaheuristic is fully assessed in both synthetic and real-world instances. This integration allows for more accurate and effective decision-making process by considering interdependencies and the trade-off between tactical and operational criteria in a comprehensive manner. It can help identify cost-saving opportunities and optimize resource allocation for these three decisions. The proposed methods can be updated and re-optimized as new data becomes available or when operational circumstances change, ensuring efficiency and performance.

This work is organized as follows. First, in Section 2, the related literature is discussed. We present the problem under study and the proposed mathematical model in Section 3. The proposed solution method is described in Section 4. We provide an extensive computational experience in Section 5. Finally, some conclusions and future research lines are discussed in Section 6.

2. Literature review

According to de Koster et al. (2007); Theys et al. (2010), the order picking incurs from 55% to 65% of the warehouse costs. Some of the operational costs involved in this activity are related to three main components: *traveling time, time to pick an item,* and *time for remaining picking activities*; being the traveling time the most costly among them (Tompkins et al. 2010; Dukic and Opetuk 2012). The three most important decisions that can be identified due to the impact on the picking travel effort are: Storage location of items, grouping customer orders to be performed in a single route, and the sequence of picker routes. In most of the literature, these decisions are studied independently or only two of them are combined. This work addresses the simultaneous integration and optimization of these three activities for a picker-to-part system in a manual warehouse. For the optimization of activities performed in automated warehouses, we refer to the reader to the works by Wang et al. (2022); Zhen et al. (2023).

We briefly describe some aspects of each activity. Then, we provide a literature review related to the integration of two or more activities in order to minimize picking costs.

2.1. *The Storage location assignment, order batching, and order picking problems*

The Storage Location Assignment Problem (Hausman et al. 1976, SLAP) is a tactical problem that decides where to place items into storage locations of a warehouse to make better use of the physical space. A general review about the SLAP can be found in Rojas Reyes et al. (2019). Since the SLAP is considered $\mathcal{NP}$-hard (Loiola et al. 2007), it is common to identify several policies used to solve the problem easily. Some of the most common policies are due to Silva et al. (2020); van Gils et al. (2018):

- *Random*: Items are placed in any available storage location as long as the capacities and dimensions of the locations allow it. It is considered a simple, fast, but inefficient heuristic for traditional commerce (it is more appropriate for e-commerce companies).
- *Dedicated*: Each item is assigned to a predefined and fixed storage location.
- *Rotation level*: Assigns high-turnover products to the best locations, those with the shortest distance to the depot (arrival/departure point).
- *Class-based*: First, items are grouped into classes, then those classes are assigned to a predefined area or zone, generally on a random basis.
- *By affinity*: Items that are often ordered together or have a certain level of affinity, are considered to be placed in the same storage location.
- *Closest space*: Items are placed in locations closest to the depot for efficient access and retrieval.

The Order Batching Problem (OBP) is the problem of grouping customers' orders into batches that will be collected later by pickers to minimize operational costs. While minimizing routing costs is an important aspect of the OBP, the objective often encompasses a broader range of operational considerations within the distribution center, with a focus on overall efficiency, effectiveness, and responsiveness to dynamic operational demands (for example, balancing workload, meeting delivery deadlines, efficient use of available resources, etc. (Won and Olafsson 2005)). Order batching reduces efforts by grouping several orders into one route, such that the better the order assignment the better the picking solutions. This can be achieved by considering several orders with similar products or by applying a priority criterion. A survey of applications of the OBP can be found in Henn et al. (2012). Since the OBP is also an $\mathcal{NP}$-hard problem (Gademann and Velde 2005), several simple rules or policies have been implemented. A common classification found in the literature is the one proposed by de Koster et al. (2007), which is as follows.

- *Basic methods*: Composed of constructive heuristics. The most widely used rule is the *First-Come-First-Serve* (FCFS) strategy, which groups orders into batches according to their arrival.
- *Seed methods*: A *seed order* is selected as a reference for other orders to be added in a specific batch. When a new order is added, it must not exceed the batch capacity, and the order to be selected from all the feasible ones, is the order that minimizes a measure of *closeness* to the *seed*. This process is repeated until there are no orders to add.
- *Savings methods*: Start by locating one order in one batch, then the number of batches is reduced by merging them as long as savings are found. This idea is similar to the one proposed by Clarke and Wright (1964), which corresponds to one of the best-known heuristics used for vehicle routing problems.

The Order Picking Problem (OPP) is the main activity of this study, and involves several tasks. The most important and time-consuming is to find the best design of routes, through a sequence of visited racks to retrieve the items, which minimizes the total traveling cost. This special task can be interpreted as a Vehicle Routing Problem (Braekers et al. 2016) or a Traveling Salesman Problem (Laporte 1992) and, in both cases, it is an $\mathcal{NP}$-hard problem. The reader is referred to the following OPP survey for more details (van Gils et al. 2018; Masae et al. 2020).

There are several ways to determine the sequence of the order picking activity considering different policies proposed for the OPP. Some of the most common routing policies are (Petersen and Aase 2004; van Gils et al. 2018):

- *Return*: Pickers arrive and depart through the same point of a warehouse aisle (only if there are items to pick).
- *S-Shape*: Pickers enter an aisle, traverse it until the end, and depart from that opposite point (in an S shape).
- *Mid-point*: Pickers enter and leave from the same point of the aisle, and only arrive as far as the middle point of it.
- *Largest gap*: Pickers enter and return to the same point of an aisle, but travel only as far as the larger gap between two adjacent items to be collected.
- *Optimal*: Pickers perform the optimal sequence (the one that provides the best solution cost), regardless of the layout design and the location of items to collect.
- *Metaheuristic*: These policies use optimization heuristic procedures to compute

the sequence of picking routes.

Current research works have studied and determined that integrating two or more of these decisions (SLA, OBP, OPP) can result in better order picking solutions, although it increases the complexity of the problem. Below, we provide a brief description of some of them.

2.2. *Integrating tactical and operational activities for the OPP*

The works presented in this section are an example of the effort and interest in improving the picking activity through the integration with other closely related decisions. These are briefly described below (see Table 1 for a comparison among them).

Hsieh and Huang (2011) develop a simulation to assess the integration of the storage location assignment (SLA) of products, order batching (OB), and picking routing (PR). Authors implement two well-known SLA and three PR policies from the literature and proposed two new OB heuristics based on k-means and self-organization map algorithms (both correspond to clustering methods, where similar customer orders are grouped into one batch). Computational experiments were carried out on generated instances of an illustrative example with up to 300 orders and 20 SKUs (using a warehouse of 400 storage locations) to assess the effect of each applied strategy. Other works that consider simulation to evaluate the integration of SLA, OB and PR policies on multi-block warehouses are proposed by Chen et al. (2010) and Chackelson et al. (2013).

A statistical analysis to evaluate the significance among several activities inside of a warehouse, which affects the order picking, was provided by van Gils et al. (2018). They applied a full-factorial ANOVA considering several policies for the order batching, storage location, zone picking, and picking routing. They found that all the hypotheses they provided were accepted, which means that combinations of these activities provide a significant effect in order picking costs. van Gils et al. (2019) extend this analysis by taking into account real-life features, e.g., picker blocking, safety constraints, and high-level storage locations.

A mathematical model and an Iterated Local Search (ILS) to formulate and solve the integration of OB, RP, and picker scheduling (sequence of the batches assigned to available pickers) considering order due times to increase the order picking efficiency were proposed by van Gils et al. (2019). Three SLA policies are used to obtain a variety of initial assignments. Computational experience and a full factorial design were carried out on synthetic and real-world instances of an automotive firm to compare exact and heuristic solutions as well as the ILS performance. Results show significant improvements when the integrated problem is considered.

Wang et al. (2020) study the optimization of the SLAP to improve the OP distance (considering an S-shape routing policy to compute it). The authors propose a data-based approach that first assigns items to aisles and then items to racks. Subsequently, the initial assignment is improved by swapping pairs of items. They assume that only one type of product must be assigned to each location, therefore, the number of items must be less or equal to the number of storage locations. Also, only one order can be fulfilled per route (no order batching is considered), and the picking cart capacity is unlimited. Extensive computational experience was provided on real-world data, solving instances with at most 4,000 products. Results show that the proposed solution algorithm outperforms other methods proposed in the literature. Silva et al. (2020) show that the integration of the SLAP and OP activities may incur in significant improvements. They propose several non-linear programming models for four cases of integration, which were linearized. The aim is to minimize the total routing costs considering each product is assigned to exactly one location and each location can have no more than one product. Computational results showed that it is not practical to solve even the small instances to optimality in less than 7200 seconds (3-5 picks, 10-60 slots). Thus, they developed a General Variable Neighborhood Search metaheuristic to obtain good-quality solutions in better computing times. Authors do not assume any specific warehouse system and apply four routing policies in their solution method.

Kübler et al. (2020) propose an iterative solution method that integrates a dynamic SLA with the OB, and the PR for the OPP. First, the dynamic SLA is obtained by solving a MILP, which considers multiple periods and forecasting techniques to evaluate the location of products (only one product per location) in the current and future time periods. Then, the OB and the PR are solved jointly through a proposed heuristic based on Particle Swarm Optimization (PSO) with evolutionary computation. Both approaches, the dynamic SLA and the PSO, are combined to solve the integrated problem. The relocation of products to storage locations for future time periods is carried out considering forecasted demands. Authors performed computational experiments on generated instances with a number of orders from 50 to 200 (2-10 items/order lines per each customer order) and 6000 products to be initially located in a multi-block warehouse of 7200 storage spots, showing good performance on those cases. Another work that proposed an approach to optimize the integration of the SLA, OB, and PR was proposed in Scholz and Wäscher (2017), where a mathematical model and an ILS to solve the OBP and the OPP simultaneously, are provided. Several well-known exact/heuristic PR policies were compared to evaluate the performance of the ILS in a two-block warehouse. The SLA is initially determined by using random and class-based policies. Assumptions such as not partitioning batches and only one product can be assigned to each storage location are considered. Results show that the ILS combined with an exact method for PR outperforms the ILS with known (heuristic) routing policies; however, for very large instances the latter is recommended.

Wagner and Mönch (2023) study the improvement of the picking activity when order batching decisions, considering a heterogeneous fleet of vehicles, are integrated. They proposed a MILP formulation and a Variable Neighborhood Search (VNS) metaheuristic to solve a case study for a company. Authors carry out extensive computational tests on large pseudo and real-world instances and compare their results with a current heuristic method that the company uses. They confirm that their proposed solution methods are efficient and, by integrating order batching decisions, companies can obtain better improvements in their warehouse operations. Other works that study the integration of the OBP and the OPP can be found in Albareda-Sambola et al. (2009); Henn and Wäscher (2012); Öncan (2015); Cao et al. (2023). Finally, some other research is focused on the integration of the OPP with other important activities such as inventory replenishment (Celik et al. 2022) and last-mile deliveries (Chen et al. 2022).

Table 1.: Related literature about the study of the three main decisions.

Literature	Main decisions			Criteria	
	SLA	OB	PR	Objective	Solution approach
Albareda-Sambola et al. (2009)	*	✓	*	Minimize picking time	Variable Neighborhood Search
Chen et al. (2010)	*	*	*	Best combination of policies	Simulation
Hsieh and Huang (2011)	*	✓	*	Best travel time, average vehicle utility, and running time	Simulation and two new OB heuristics.
Henn and Wäscher (2012)	*	✓	*	Minimize total distance	Tabu Search and an Attribute-based hill climber metaheuristics.
Chackelson et al. (2013)	*	*	*	Best total picking time and order maturity time	Simulation
Öncan (2015)	*	✓	*	Minimize total traveled distance	Three MILPs, an Iterated Local Search.
Scholz and Wäscher (2017)	*	✓	✓	Minimize the total tour length	MILP, Iterated Local Search
van Gils et al. (2018)	*	*	*	Best overall picking performance	Statistical analysis
van Gils et al. (2019)	*	*	*	Best travel time in terms of picker blocking	Simulation and Statistical analysis
van Gils et al. (2019)	*	✓	✓	Maximize order picking efficiency	A MILP and a metaheuristic.
Wang et al. (2020)	✓	*	*	Best storage location that minimize the total travel distance	Integer programming and a Data-based approach.
Kübler et al. (2020)	✓	✓	✓	Best storage location, batching, and routing that minimize the total travel distance	MILP for SLAP and Particle Swarm Optimization for OB and PR.
Chen et al. (2022)	*	✓	✓	Minimize operation costs and overdue penalty costs	Two-stage Iterated Search
Cao et al. (2023)	*	✓	✓	Minimize the total picking time and a penalty for due dates of orders	Non-linear programming, Iterated Local Search
Wagner and Mönch (2023)	*	✓	✓	Minimize the total travel distance	MILP, Variable Neighborhood Search
This work	✓	✓	✓	Minimize total picking costs	MILP, Iterated Local Search

* Solved by applying a predetermined policy.
✓ Solved by using optimization methods.

In this work, we study an OPP that integrates, not only the storage location of products and the sequence of picker routes but also order batching decisions, simultaneously. We proposed a mathematical model and an IGLS metaheuristic to obtain good-quality solutions for a real-world case study of a Mexican retail firm. The mathematical model is formulated as a mixed-integer linear program that can be used to obtain feasible solutions for small (but still real-world size) instances using a general-purpose solver and, to the best of our knowledge, it can be considered the first mathematical formulation that integrates these three main decisions. On the other hand, the proposed IGLS provides a good alternative to solve the problem, without considerably reducing the quality of solutions, in reasonable computing times for large-scale instances.

Moreover, most of the previous works mentioned above, specifically those that study the integration of these three decisions (see Table 1), only consider the optimization of one or two decisions simultaneously, generating lower quality solutions than those obtained when the three decisions are integrated and optimized. Moreover, the works from the literature consider several assumptions that are tackled in this work in a different manner, e.g., that only one product type can be assigned to each storage location (we consider a maximum affinity value to determine what product types, and their quantities, can be assigned together), the absence of due times for customer orders (our problem considers a maximum processing time for each one), and the use of an exact method to obtain each final picker route embedded in the proposed heuristic approach, which allows not to depend completely on the type of layout to be used to obtain good routes, giving the opportunity to use the same approach regardless the design of the warehouse.

3. Problem definition and mathematical model

3.1. *Problem definition*

Given a directed graph $G = (N, A)$, with $N = L \cup \{0\}$ as the set of nodes formed by a set L of storage locations/racks and the depot (node 0, from which pickers depart and return after collecting the customer orders), and a set A of available arcs between each pair of storage locations (where an "arc" means any direct path between two adjacent or nearby racks that does not involve passing through another intermediary). Let P, O, W, and B be the sets for products, customer orders, pickers, and batches, respectively. Furthermore, each unit of product $p \in P$ has a volume V_p and a deterministic demand $D_{po} \geq 0$ required for a given order $o \in O$. Q^L and Q^W correspond to the homogeneous capacities for racks and picker carts, respectively. Any pair of products $(p, q) \in P$, assigned to the same rack, must not exceed a level of affinity $\gamma \in [0, 1]$. In addition, each order $o \in O$ must be collected before a maximum processing time $T_o^{\max}$ to ensure that orders are ready for delivery. Finally, a cost $c_e \geq 0$, a traveling time $t_e \geq 0$, and a distance $d_e \geq 0$, per each arc $e \in A$, are defined.

Then, the aim of the order picking problem, called from now on the *PSB*, which considers storage location, order batching, and picker routing decisions is to determine where to locate products into racks, orders into batches, and the sequence of picker routes, in order to minimize the total picking cost.

Some other assumptions must be considered:

- Layout aisles are wide enough, i.e., pickers can go back and forth through the aisles.
- Customer orders must not be partitioned (each batch consists of one or more complete orders). Perfect order information is assumed.
- Products can be placed in one or more storage locations (if necessary).
- The volume of the products is equivalent to the space occupied by their packaging.
- A Fishbone layout with a single depot is considered. The main reason to use this non-traditional layout is because the firm seeks to migrate to this design in the

future. In addition, it has been shown some advantages over traditional layouts (Cardona et al. 2012; Dukic and Opetuk 2012).
- Acceleration/deceleration not considered during the picking process. Picker blocking and congestion are not considered.

3.1.1. Product affinity

An important criterion considered by the firm is the affinity of products to be located in the racks. The motivation of doing this is because by placing items with high affinity near each other, pickers can easily locate and retrieve multiple related items in a single trip, minimizing unnecessary movement and, consequently, improving the order picking activity. This affinity can be measured in several manners (Kofler et al. 2014). However, the firm has its own procedure to compute the affinity between two products p and q, which is explained below.

- **Step 1.** Each available product p has an affinity code assigned to it, which corresponds to an alphanumeric value for each product category defined by the firm (see an example in Table 2). Products (Stock Keeping Units, SKUs) from the same category have the same affinity code, for example P4 and P6 (column 2). The codes are sorted alphabetically and an ordinal value is assigned for each unique category (column 3).

Table 2.: Affinity values

Product	Affinity code	Value
P2	0A10800	1
P1	0A13400	2
P6	0A13401	3
P4	0A13401	3
P3	0B11200	4
P5	0B13300	5

- **Step 2.** Given the data provided above, the affinity value for any pair of products p and q, is computed as follows.

$$\texttt{Aff}(p,q) = \frac{|\texttt{Val}_p - \texttt{Val}_q|}{n_{cat}},$$

where $\texttt{Val}_p$ and $\texttt{Val}_q$ are the corresponding individual values provided in the third column of Table 2, and n_{cat} is the number of the resulting product categories. Therefore, $\texttt{Aff}(p,q)$ resulted in a [0,1) value, where a value near 0 represents a pair of similar products and, a value near 1, represents two dissimilar products. This $\texttt{Aff}(p,q)$ is computed for each possible pairwise (p,q) to generate an affinity matrix (Table 3), which will be useful to determine the products that can be placed in the racks.

Table 3.: Affinity matrix

	P1	**P2**	**P3**	**P4**	**P5**	**P6**
P1	x	0.20	0.40	0.20	0.60	0.20
P2	-	x	0.60	0.40	0.80	0.40
P3	-	-	x	0.20	0.20	0.20
P4	-	-	-	x	0.40	0.00
P5	-	-	-	-	x	0.40
P6	-	-	-	-	-	x

Finally, the affinity $\texttt{Aff}(p,q)$ is usually limited to a γ value, which corresponds to the maximum dissimilarity allowed between products located in the same rack.

3.2. *Mathematical model*

In this section, we introduce a mixed-integer linear programming (MILP) model for the PSB problem. This model considers a set of decision variables for each studied activity (the storage assignment, order batching, and the pickers' routing). The parameters, sets, and variables defined for this MILP are described below.

Sets and parameters

- L, A, P, O, W, B: Sets for storage locations (racks), available arcs (connections between racks), products, orders, pickers, and batches, respectively.
- $G = (N, A)$: A directed graph with $N = L \cup \{0\}$ nodes (0 as the depot) and A as the available arcs.
- A_l^+/A_l^-: Set of arcs that enter/exit to/from a location $l \in L \cup \{0\}$, where $A_l^+, A_l^- \subseteq A$.
- V_p: Volume of product $p \in P$.
- $\gamma \in [0,1]$: A maximum value of dissimilarity (affinity level) allowed between any two (p,q) products located in the same rack.
- Q^L, Q^W: Maximum capacity (volume) of racks and carts, respectively.
- D_{po}: Demand of product $p \in P$ in a given customer order $o \in O$.
- $T_o^{\max}$: Maximum processing time of an order $o \in O$.
- M: An arbitrarily large number.
- c_e: Picking cost, traveling time, and distance for each arc $e \in A$.

Decision variables for the storage location of products

- $y_p^l = \begin{cases} 1 & \text{If product } p \in P \text{ is assigned to location } l \in L \\ 0 & \text{Otherwise} \end{cases}$
- $a_p^l \in \mathbb{Z}^+$: Number of units of product $p \in P$ assigned to location $l \in L$.

Decision variables for the order batching

- $b_{oi}^w = \begin{cases} 1 & \text{If order } o \in O \text{ is assigned to batch } i \in B \text{ and picker } w \in W \\ 0 & \text{Otherwise} \end{cases}$
- $\tau_{oi}^{ew} = \begin{cases} 1 & \text{If arc } e \in A \text{ is traversed by picker } w \in W \text{ to collect order } o \in O \\ & \text{assigned to batch } i \in B \\ 0 & \text{Otherwise} \end{cases}$

Decision variables for the order picking

- z_{pol}^{iw}: Binary variables equal to 1 if product $p \in P$, from order $o \in O$ of batch

$i \in B$ and located at $l \in L$, is collected by picker $w \in W$, and equal to 0 otherwise.

- r_e^{iw}: Binary variables equal to 1 if arc $e \in A$ is traversed by picker $w \in W$ while collecting orders of batch $i \in B$, and equal to 0 otherwise.
- $f_e^{iw} \geq 0$: Non-negative and continuous variables representing the load of the cart used by picker $w \in W$ when traversing arc $e \in A$ to fulfill batch $i \in B$.
- $q_{pol}^{iw} \in \mathbb{Z}^+$: Number of units of product $p \in P$ collected by picker $w \in W$ at rack $l \in L$, for any order $o \in O$ that belongs to batch $i \in B$.

Then, the PSB is defined as follows.

$$\min \quad \sum_{i\in B}\sum_{w\in W}\sum_{e\in A} c_e r_e^{iw} \tag{1}$$

$$\text{s.t.} \quad \sum_{l\in L} y_p^l \leq \frac{\sum_{o\in O} V_p D_{po}}{Q^L} \qquad p \in P \tag{2}$$

$$\sum_{l\in L} a_p^l = \sum_{o\in O} D_{po} \qquad p \in P \tag{3}$$

$$V_p a_p^l \leq Q^L y_p^l \qquad p \in P, l \in L \tag{4}$$

$$\sum_{p\in P} V_p a_p^l \leq Q^L \qquad l \in L \tag{5}$$

$$y_p^l + y_q^l \leq 1 \qquad p, q \in P | \texttt{Aff}(p,q) > \gamma, l \in L \tag{6}$$

$$\sum_{w\in W}\sum_{i\in B}\sum_{o\in O} z_{pol}^{iw} \leq M y_p^l \qquad p \in P, l \in L \tag{7}$$

$$V_p q_{pol}^{iw} \leq Q^W z_{pol}^{iw} \qquad p \in P, o \in O, i \in B, w \in W, l \in L \tag{8}$$

$$\sum_{l\in L}\sum_{o\in O}\sum_{p\in P} V_p q_{pol}^{iw} \leq Q^W \qquad i \in B, w \in W \tag{9}$$

$$\sum_{i\in B}\sum_{w\in W}\sum_{o\in O} q_{pol}^{iw} \leq a_p^l \qquad p \in P, \in L \tag{10}$$

$$\sum_{l\in L}\sum_{w\in W}\sum_{i\in B} q_{pol}^{iw} = D_{po} \qquad p \in P, o \in O \tag{11}$$

$$\sum_{e\in A_l^+} r_e^{iw} \leq 1 \qquad l \in L, i \in B, w \in W \tag{12}$$

$$\sum_{w\in W}\sum_{e\in A_0^+} r_e^{iw} = 1 \qquad i \in B \tag{13}$$

$$\sum_{e\in A_l^+} r_e^{iw} = \sum_{e\in A_l^-} r_e^{iw} \qquad i \in B, w \in W, l \in L \tag{14}$$

$$\sum_{e\in A_l^+} f_e^{iw} = \sum_{e\in A_l^-} f_e^{iw} + \sum_{o\in O}\sum_{p\in P} V_p q_{pol}^{iw} \qquad w \in W, i \in B, l \in L \tag{15}$$

$$f_e^{iw} \leq Q^W r_e^{iw} \qquad e \in A, i \in B, w \in W \tag{16}$$

$$\sum_{i\in B}\sum_{w\in W}\sum_{e\in A_0^-} f_e^{iw} = 0 \tag{17}$$

$$b_{oi}^w \leq \sum_{l\in L}\sum_{p\in P} z_{pol}^{iw} \leq M b_{oi}^w \qquad o \in O, i \in B, w \in W \tag{18}$$

$$\sum_{w\in W}\sum_{i\in B} b_{oi}^w = 1 \qquad o \in O \tag{19}$$

$$r_e^{iw} + b_{oi}^w - 1 \leq M\tau_{oi}^{ew} \qquad o \in O, w \in W, i \in B, e \in A \tag{20}$$

$$\tau_{oi}^{ew} \leq r_e^{iw} \qquad o \in O, w \in W, i \in B, e \in A \tag{21}$$

$$\tau_{oi}^{ew} \leq b_{oi}^w \qquad o \in O, w \in W, i \in B, e \in A \tag{22}$$

$$\sum_{i\in B}\sum_{e\in A} t_e \tau_{oi}^{ew} \leq T_o^{\max} \qquad o \in O, w \in W \tag{23}$$

$$b_{oi}^w \leq b_{oi}^{w-1} \qquad o \in O, i \in B, 2 \leq w \leq |W| \tag{24}$$

$$y_p^l \in \{0,1\}, a_p^l \in \mathbb{Z}^+ \qquad p \in P, l \in L \tag{25}$$

$$z_{pol}^{iw} \in \{0,1\}, q_{pol}^{iw} \in \mathbb{Z}^+ \qquad p \in P, o \in O, w \in W, i \in B, l \in L \tag{26}$$

$$r_{ie}^w \in \{0,1\}, f_{ie}^w \geq 0 \qquad e \in A, i \in B, w \in W \tag{27}$$

$$b_{oi}^w \in \{0,1\} \qquad o \in O, i \in B, w \in W \tag{28}$$

$$\tau_{oi}^{ew} \in \{0,1\} \qquad o \in O, i \in B, w \in W, e \in A \tag{29}$$

$$h_o \in \{0,1\} \qquad o \in O \tag{30}$$

The objective function (1) minimizes the total picking routing cost. Constraints (2)-(11), (12)-(17), and (18)-(23), correspond to the storage location, picking routing, and batching constraints, respectively. More specifically, Constraints (2) specify the maximum number of locations where a product can be assigned. Constraints (3), determine that the total quantity assigned to racks, for a given product p, must satisfy at least its total demand. Constraints (4)-(5) guarantee that the volume of the quantity of product delivered at each location must not exceed the rack capacity. Constraints (6) avoid to place together (in the same rack) two products whose affinity level does not exceed a γ value. Consequently, these constraints reduce the number of products stored in multiple locations while providing a flexible design that can deal with demand patterns and variance for different scenarios. Constraints (7) link variables $\mathbf{y}$ and $\mathbf{z}$, and constrain $\mathbf{z}$ when no product p is assigned to rack l. Constraints (8)-(9) specify that the total product volume collected by a picker must not exceed the cart capacity. Constraints (10) prevent collecting more than the quantity available, for a given product p, at location l. Constraints (11) establish that the total quantity of product p, for a given order o, must be equal to the required demand. Constraints (12) determine that at most one cart must arrive at location l per each batch i and picker w. Constraints (13) specify that each batch i must be assigned to a single picker. Constraints (14) define that the number of carts that arrive and depart, at/from a specific location, is the same for a given batch and picker. Constraints (15) establish that the total load of each picker cart, when arriving at a rack l, must be the volume previously loaded plus the volume of the quantity to be picked up at that location (also avoid subtours). Constraints (16)-(17), prevent to exceed the cart capacity and specify that each cart must depart with empty load from the depot, respectively. Constraints (18)-(22) assign orders to batches and batches to pickers. Constraints (23) establish that the total time required to complete an order must be less than its maximum processing time. It can be noticed that Constraints (23) determine a certain level of batching precedence when a picker performs more than one route. Constraints (24) are valid inequalities to break the symmetry of the pickers' carts assignment. Constraints (25)-(30) represent the domain of variables.

Computational complexity: The PSB is an order picking problem that considers assignment, batching, and routing decisions. We can demonstrate that the PSB is $\mathcal{NP}$-hard as follows. First, PSB is clearly in $\mathcal{NP}$ since feasibility can be verified in polynomial time. Now, we will show that the TSP, a well-known $\mathcal{NP}$-hard problem, is polynomially reducible to the PSB. For this, we consider a particular case of the PSB when the number of available racks is equal to the number of products $|L| = |P|$ (all products have the same volume $V_p = 1$), there is only one customer order $|O| = 1$ that requires only 1 unit per product of the list $D_{po} = 1$, $p \in P$; one picker is available to perform the route and, since there is only one customer order and it is not possible to partition it, only one batch must be performed during the day, so $|W| = |B| = 1$; racks and carts capacities are $Q^L = 1$ and $Q^W = \infty$, respectively. There is no possibility to assign more than one product at each rack since $\gamma = 0.0$ and racks capacity does not allow it; also, $T_o^{\max} = \infty$ (there is enough time to process the order). For this special case, the PSB is reduced to find the minimum tour that visits all racks in $L \cup \{0\}$ (departing from the depot) resulting in a TSP, which in turn is $\mathcal{NP}$-hard (Gutin and Punnen 2007). Therefore, the PSB is at least as hard as the TSP and, given the size of the real-world instances, we propose a metaheuristic to obtain good quality solutions in reasonable time.

4. The Iterated Greedy Local Search for the PSB

Since obtaining the optimal solution for real-world problems in this context can be a hard task for exact methods, we propose an Iterated Greedy Local Search (Lourenço et al. 2003) metaheuristic for the PSB (IGPSB). The proposed IGPSB (see Algorithm 1) starts by generating an initial feasible solution s constructed by solving a MILP/heuristic to solve the storage location subproblem, a first-fit heuristic to obtain the assignment of orders to batches and batches to pickers, and finally solving a TSP for each batch to obtain the sequence of picker routes. Then, s is improved by applying two local search (LS) procedures. Afterwards, a second phase is performed, where s is partially *destroyed* and greedily *reconstructed* to be, then, improved by the LSs. Therefore, if the solution value $f(s)$, after reconstruction, is better than $f(s')$, then s replaces s' and, if $f(s)$ is also better than $f(s^*)$, s^* is updated. Otherwise, if $f(s)$ is worse than $f(s')$, an evaluation procedure is performed to decide whether or not to accept s and replace s'. The IGPSB stops when certain criteria are met and the best solution found s^* is provided.

Algorithm 1 IGPBS

```
Input: : $L$ (racks), $P$ (products), $W$ (pickers), $O$ (orders),
    $\gamma$ (max. affinity value), %Dest (% solution destruction),
    $\beta$ (factor for worse solution acceptance).
Output: $s^*$: Best solution found.
                                        ▷ Initial solution
 1: $s \leftarrow$ getInitialSolution($L, P, W, O, \gamma$) ;
 2: $s \leftarrow$ LocalSearch($s$)
                                        ▷ Iterative phase
 3: $s' \leftarrow s$, $s^* \leftarrow s$
 4: while stopping criteria not met do
 5:     $s'' \leftarrow$ Destruction($s$, %Dest)
 6:     $s \leftarrow$ reConstruction($s''$, %Dest)
 7:     $s \leftarrow$ LocalSearch($s$)
 8:     if $f(s) < f(s')$ then
 9:         $s' \leftarrow s$
10:         if $f(s) < f(s^*)$ then
11:             $s^* \leftarrow s$
12:         end if
13:     else
14:         {s, $s'$} $\leftarrow$ evaluateWorseSolution($s, s', s^*, \beta$)
15:     end if
16: end while
17: return $s^*$
```

The main components of the IGPSB are described in more detailed below.

4.1. *Phase 1: Generation of an initial solution*

In the first phase of the IGPSB, an initial feasible solution s is obtained. We propose a simple constructive heuristic composed of three steps. Finally, the resulting s is improved by two local searches proposed for this problem.

4.1.1. Constructive heuristic

In the first step of the constructive heuristic, the storage location of products to racks is determined. Then, in the second step, the order batching is obtained. Once these two decisions are performed, the third step consist of solving a TSP to obtain each picker route. Further details are provided below.

- *Step 1a (for small-medium instances):* To obtain the storage location assignment (SLA) decisions, the MILP subproblem (31)-(33), is solved. This step is only performed when small- or medium-scale instances are evaluated.

  Let $\bar{y}_l \in \{0,1\} = 1$ if rack l contains product p; otherwise, 0. Then, the subproblem that obtains a distribution plan to locate products in racks is as follows.

$$\min \quad \sum_{l \in L} c_{0l} \bar{y}_l \tag{31}$$

$$
\begin{aligned}
\text{s.t.}\quad & \bar{y}_l \leq \sum_{p\in P} y_p^l \leq M\bar{y}_l && l \in L && (32)\\
& (2) - (6) &&&&\\
& (25) &&&&\\
& \bar{y}_l \in \{0,1\} && l \in L && (33)
\end{aligned}
$$

The objective function (31) aims to minimize the distance between any used rack and the depot (closest space policy). The new set of constraints (32), strongly constrains variables $\bar{y}_l$ to take no value if there is no product assigned to a storage location l. Constraints (33) delimit the domain of the new decision variables.

- *Step 1b (for large instances):* The previous step has the limitation that, when $|P|$ is large (greater than 2000), it is difficult to be solved by a general-purpose solver. Therefore, no solution for the SLA subproblem can be created, showing a limitation to perform the further phase of the IGPSB. As an alternative to address this issue, we have proposed a simple heuristic to create an initial assignment of products into racks and thus continue with the batching and routing decisions to generate a first initial feasible solution for the IGPSB. The proposed heuristic, shown in Algorithm 2, starts by sorting the set of products and available racks according to their affinity and their closeness to the depot, respectively. Then, in a sequential order, the first rack l is *open* and the first product p is assigned to l if there exists available space to assign (at least) a proportion of its total demand, taking into account that its affinity value, with respect to the first product $\bar{p}$ assigned to l, must be less than γ; otherwise, the current rack is no longer considered and another one must be open to locate the products that are not yet assigned. Once the total units of products p are place into racks, the product is removed from the non-assigned set. Finally, a feasible assignment is provided if all products were located in a given storage location.

Algorithm 2 `SLAHeuristic()`.

```
Input: : $L, P, D_p^T, Q^L$        ▷ Set of racks, products, total demand per product, rack capacity
Output: As                                                               ▷ Final assignment

1: $P^s \leftarrow$ sortByAffinity($P$)                                    ▷ $P^s$ is sorted $P$
2: $L^s \leftarrow$ sortByClosenessToDepot($L$)                            ▷ $L^s$ is sorted $L$
3: $Q^L_{rem} \leftarrow Q^L$                                             ▷ Remaining capacity
4: while $P^s \neq \emptyset$ and $L^s \neq \emptyset$ do
5:     Select the first rack $l \in L^s$
6:     Select the first item $p \in P^s$
7:     Set $\bar{p}$ the first item $p$ assigned to rack $l$
8:     $D_{cov} \leftarrow \min(D_p^T, Q^L_{rem})$                          ▷ Covered demand
9:     if $D_{cov} > 0$ and $\texttt{Aff}_{\bar{p}p} \leq \gamma$ then
10:        $As(l, p, D_{cov})$: Assign $D_{cov}$ units of $p$ to $l$
11:        $Q^L_{rem} \leftarrow Q^L_{rem} - D_{cov}$
12:        $D_p^T \leftarrow D_p^T - D_{cov}$
13:    else
14:        Remove $l$ from $L^s$
15:        Remove $p$ from $P^s$ if $D_p^T$ is equal to 0
16:        $Q^L_{rem} \leftarrow Q^L$
17:    end if
18: end while
19: if $P^s \neq \emptyset$ then
20:    $As(l, p, q) \leftarrow \emptyset$                                    ▷ Infeasible, no assignment found
21: end if

22: return As
```

- *Step 2:* Once the SLA subproblem is solved. We obtain the order batching assignment by adapting a first-fit algorithm (Lodi et al. 2002), considering the following steps (see Algorithm 3):
  a) Sort customer orders in a non-decreasing order according to their maximum processing time $T_o^{\max}$ (line 1). The reasoning behind this is to assign, in the first batches carried out by pickers, orders that have less $T_o^{\max}$ available to be completed (priority policy).
  b) Given an open batch $\mathcal{B}$, select the first order of the sorted set and assign it to $\mathcal{B}$ as long as the remaining capacity of the batch (corresponding to the picker carts, Q^W) and the $T_o^{\max}$ for all its orders are not exceeded. Otherwise, the current batch $\mathcal{B}$ is closed and it is assigned to a picker w. Then, a new batch $\mathcal{B}$ is opened, and the current order of the list is assigned to the new $\mathcal{B}$ if possible. The algorithm iterates until there are no orders to assign (lines 4-21). Notice that, since each picker can only perform one batch per route, the maximum number of batches that can be fulfilled simultaneously is $|W|$; nevertheless, pickers can carried out more than one route (batch) sequentially during the day.

Algorithm 3 `getBatching`

```
Input: : $O, Q^W$, **As**                                         ▷ Orders, total card capacity, location assignment
Output: $B_w$                                                     ▷ Final batching assignment per picker

 1: $\hat{O} \leftarrow$ sortByTmax($O$)                          ▷ $O$ is sorted by $T_o^{\max}$
 2: $B_w \leftarrow \emptyset \qquad w \in W$
 3: $w \leftarrow 1$
 4: while $\hat{O} \neq \emptyset$ do
 5:     $Q^W_{rem} \leftarrow Q^W$                                ▷ Remaining cart capacity
 6:     Open a new batch $\mathcal{B}$
 7:     for $o \in \hat{O}$ do
 8:         $Vol_o \leftarrow$ getTotalVol($o$)
 9:         if $Vol_o \leq Q^W_{rem}$ and TotRouteTime($B_w, \mathcal{B} \cup o$, **As**) $\leq$ minTmax($\mathcal{B}$) then
10:             $\mathcal{B} \leftarrow \mathcal{B} \cup o$
11:             $\hat{O} \leftarrow \hat{O} \setminus o$
12:             $Q^W_{rem} \leftarrow Q^W_{rem} - Vol_o$
13:         else
14:             $B_w \leftarrow B_w \cup \mathcal{B}$
15:             $w \leftarrow w + 1$
16:         end if
17:     end for
18:     if $w > |W|$ then
19:         $w \leftarrow 1$
20:     end if
21: end while

22: return $B_w$
```

- *Step 3:* Once it is known where products are located, and how orders are grouped in batches, a TSP for each picker route (batch) is solved. Since we are working on an incomplete graph, i.e., there is no a direct path to connect several pairs of locations (l_1, l_2), we simplify this by transforming the current graph into a complete one using the Floyd-Warshall algorithm (Floyd 1962). Then, the classic well-known TSP is performed to obtain the sequence of picker routes.

4.1.2. Local search procedures

In order to improve a given solution s, we proposed two LS procedures. The first one improves the selection of storage locations and the second one attempt to improve the order batching. The LS procedures are explained below.

Storage location LS.

The LS proposed for the storage location decisions, improves a solution s by applying a sequential neighborhood search scheme, which performs a 1-1 move that consist in selecting a rack l_1 from s and a rack l_2, which does not belong to s, and swap them (only if it is feasible and improves the solution), moving all product assigned from one rack to another (Fig. 1).

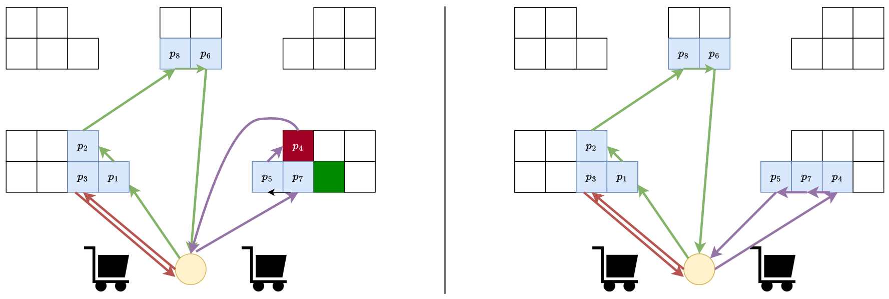


Fig. 1.: LS for storage location decision

Order batching LS.

The second LS (Fig. 2) improves the order batching by exploring a neighborhood where 1-moves are applied. In this LS, one order o_1 from batch b_1 is moved to another batch b_2 (not necessarily of a different picker). If solution remains feasible (cart capacity and processing times are not exceeded) and if its cost improves, the move is accepted (we consider a First-Found strategy).

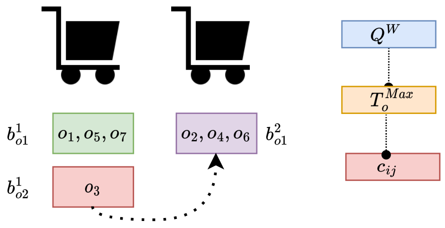


Fig. 2.: LS for order batching decision

4.2. *Phase 2: Iterative phase*

Once the initial solution is obtained, the second phase of the IGPSB starts by partially *destroying* the solution s, and then *reconstructing* it to obtain a new one, which will be improved by applying the LS procedures explained previously. If the reconstructed solution is better than the current one, this is accepted and replaces the current solution (and the incumbent one if it is the case). If the new solution is worse than the current accepted solution, a criterion of acceptance is evaluated to determine whether or not it will replace the current one in the next iteration. The IGPSB stops when a stopping criterion is met. Fig. 3 shows the main steps of this phase and each component is explained with more detail below.

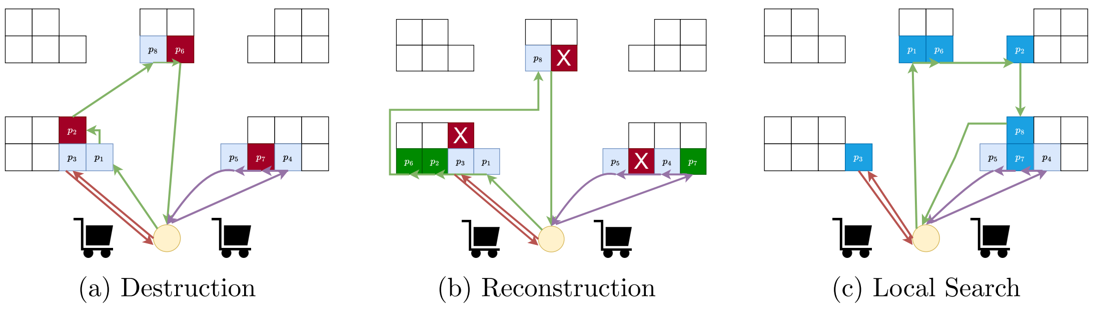


Fig. 3.: The main steps of the iterative phase of the IGPSB

Destruction-Reconstruction. During the destruction step, a given solution is perturbed by removing a proportion of it in order to be reconstructed greedily in the next step. For the studied problem, we focused the destruction on the selected racks where products are located. This procedure is as follows.

- **Destruction.** Select randomly a proportion (measured by parameter `%Dest`) of the total number of racks assigned in s, and remove them resulting in a partial solution s''. Products from these racks remain grouped as initially allocated.
- **Reconstruction.** For each rack $l \in L \setminus s''$, compute the cost between depot and l. The rack l with the cheapest cost (shortest distance) is selected to be a new element of the current s''. The groups of products assigned to the corresponding removed racks, are assigned in the same sequence as those racks were removed. Once the number of storage locations added corresponds to the proportion `%Dest`, the solution is complete.

Improvement. In this step, the solution reconstructed is improved by applying the proposed LS procedures mentioned in Section 4.1.2.

Selection. Once the solution s is improved, the algorithm must decide wheter or not this new improved solution will be accepted (Algorithm 1, lines 8-15). If the solution value $f(s)$ is better than $f(s')$, then s replaces s'. Furthermore, if $f(s)$ is better than the solution value of the incumbent $f(s^*)$, then, s will replace s^*. On the other hand, if $f(s)$ is worse than $f(s')$, s must be evaluated to determine if it will be accepted (see Algorithm 4). For this, a *Gap* value is computed and multiplied by a factor β to determine a threshold τ, which will be used to decide if a worse solution will replace s' (increasing the margin of acceptance of poorer quality solutions in subsequent iterations). Also, a counter aw (naw) and a variable faw ($fnaw$), that accumulate the number of iterations and the solution value of accepted (non-accepted) worse solutions, respectively, are defined to update the threshold τ in subsequent iterations.

Algorithm 4 `evaluateWorseSolution`(s, s', s^*, β)

.

```
Input: : $s, s', s^*, \beta$        ▷ New, current, and best solution, respectively; factor for poor-quality solution acceptance.
Output: $s, s'$        ▷ Accepted/updated solutions.

1:  $Gap \leftarrow \frac{|f(s)-f(s^*)|}{f(s^*)}$
2:  if $iter = 0$ then
3:      $\tau \leftarrow \beta \times Gap$
4:  end if
5:  if $Gap \leq \tau$ and $Gap > 0$ then
6:      $s' \leftarrow s$        ▷ $s$ is accepted, $s'$ is updated.
7:      $faw \leftarrow faw + f(s)$
8:      $aw \leftarrow aw + 1$, $naw \leftarrow 0$
9:  else
10:     $s \leftarrow s'$        ▷ $s$ is not accepted.
11:     $fnaw \leftarrow fnaw + f(s)$
12:     $naw \leftarrow naw + 1$, $aw \leftarrow 0$
13: end if
14: if $naw = 0.1 \times iterIGLS_{\max}$ then        ▷ After $naw$ non-accepted worse solutions
15:     $Gap \leftarrow \frac{|\frac{fnaw}{naw} - f(s^*)|}{f(s^*)}$
16:     $\tau \leftarrow \beta \times Gap$
17:     $fnaw \leftarrow 0$, $naw \leftarrow 0$
18: end if
19: if $aw = 0.1 \times iterIGLS_{\max}$ then        ▷ After $aw$ accepted worse solutions
20:     $Gap \leftarrow \frac{|\frac{faw}{aw} - f(s^*)|}{f(s^*)}$
21:     $\tau \leftarrow \beta \times Gap$
22:     $faw \leftarrow 0$, $aw \leftarrow 0$
23: end if

24: return $\{s, s'\}$
```

In the following section, we perform several computational experiments in order to evaluate the proposed solution algorithms.

5. Computational experience

Computational experiments were conducted on a Workstation HP Intel(R)-Xeon(R) at 3.5GHz with 64 GB RAM (Win 10 Pro, 64 bits) and a processor with 6 cores. The solution algorithms have been implemented in C++ and, in the case of the formulations, we use ILOG Concert Technology API (CPLEX 20.1), setting to only one thread per optimization. For solving the TSP during the IGPSB, we use the Concorde solver, which is a state-of-the-art branch-and-cut exact solver that can obtain solutions for large scale instances (the largest solution solved to optimality using Concorde has 85,900 nodes) and, considering the potential size of each resulting picker route, we consider it an acceptable tool to solve this subproblem.

For all tested instances, we evaluate the quality of the solution produced by a specific method M, i.e. s^M, by computing the *Relative Percentage Deviation* (RPD) of its objective function value, $f(s^M)$, with respect to the best-known solution, $f(s^*)$. The RPD is defined as

$$\text{RPD} = \frac{f(s^M) - f(s^*)}{f(s^*)} \times 100\%. \tag{34}$$

Layout. The layout used for the tests represents a fishbone layout with at most 241 storage locations (see Fig. 4), where the black square corresponds to the depot from which pickers depart. This layout is used for the real-world instances subset explained below. Other two reduced layouts with up to 58 and 121 racks (including depot), extracted according to their position shown in Fig. 4, were used for testing pseudo-real instances.

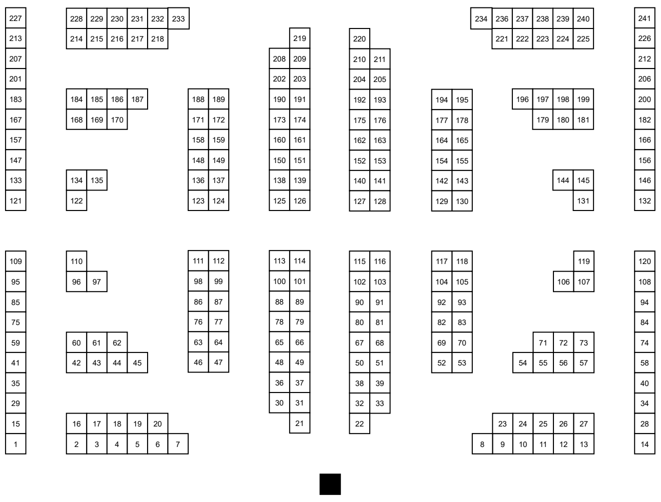


Fig. 4.: Fishbone layout

Picking costs. Costs are based on traveled distance and time, i.e., $c_e = t_e/10$ and $t_e = \alpha d_e$ (d_e and α, are the distance traveled through the arc e and a time factor per each unit traveled). For the computational experience, the firm determined a time factor of $\alpha = 1.4286$ for each unit of distance traveled and, for every 10 seconds elapsed, the cost of travel is fixed at one Mexican peso.

All instances, as well as any complementary material and results, are publicly available at https://github.com/DianaHuertaM/IGPSB.git.

This section is organized as follows. In Section 5.1, we provide a description of the generated instances. In Section 5.2, we provide a preliminary analysis of the impact of the maximum level of affinity γ for the storage location assignment decisions. Finally, Section 5.3 describes the final performance of the proposed solution method.

5.1. *Instances*

Tactical and operational decisions are evaluated by using the same set of historical customer orders processed by the firm for six months. This allows for the tightly coupled optimization of the storage location of products and batching decisions in order to improve picking costs. The model is trained on this data to identify the best setting for the operation to handle multiple scenarios required during different days of the week.

We have created two sets of instances to assess the proposed solution method and determine its performance. The description is provided below.

- **Calibration set:** This set is composed of 40 pseudo-real instances with size $r = 30, 60, 100$, and 500 order lines (where an order line represents the information of a specific item required by a given customer); the number of orders, pickers, and products corresponds to $|O| = |W| = 5$ and $|P| = 30$ (for $r = 30/60$ instances), and $|O| = |W| = 10, 20$ and $|P| = 50$ (for sizes $r = 100/500$, respectively). Products and their volume (measured in cm^3) were selected randomly from a general list provided by the company; the number of units ordered for a specific product p in a given order o is set to $D_{po} = 1$ units. Vehicle and rack capacities are set to $Q^W = 126000$ cm^3 and $Q^L = 216000$ cm^3, respectively. The maximum processing time required for a given customer order X_o is computed as $T_o^{\max} = \rho|X_o|$, where $|X_o|$ corresponds to the number of products that belong to that order and ρ determines an approximate number of seconds needed to handle each product during the picking activity; for the tests, we use a value $\rho = 100$ seconds.
- **Performance set:** This set is composed of 60 pseudo-real instances (subset A) and 14 real-world instances (subset B). We explain both sets in more detail. Vehicle and rack capacities are set to $Q^W = 1680000$ cm^3 and $Q^L = 3192000$ cm^3, respectively.
  - **Subset A:** This subset corresponds to 30 small instances (10 per size) with $r = 30$, 60, and 100 order lines; number of orders, pickers, and products of $|O| = |W| = 3, 5$ and $|P| = 20$; and 30 large instances (10 per size) with $r = 1000$, 3000, and 5000 order lines; number of orders, pickers, and products of $|O| = |W| = 150, 250$ and $|P| = 100$. The number of units ordered for a specific product was generated using a uniform distribution on the interval $\sim U[1, 3]$, and the products for each order were selected randomly for a given list of products.
  - **Subset B:** This set is composed of 13 instances obtained from the data provided by the company. The size of them varies from 362 to 8197 order lines, number of pickers $|W| = 1 - 44$, number of orders $|O| = 1 - 654$, and number of products of $|P| = 106 - 4252$. The $T_o^{\max}$ value for a given order $o \in O$ was obtained by computing the following formula:

    $$T_o^{\max} = t_{\text{tot}} + t_{\text{rem}}.$$

    where $t_{\text{tot}} = t_{\text{fin}} - t_{\text{ini}}$ (difference between the initial and final picking time in the real-word case), and $t_{\text{rem}} = 1800 - (t_{\text{tot}} \mod 1800)$, which is the remaining time to complete the nearest half hour.

To easily identify the main characteristics of both sets, we have named each instance as Inst$\mathbb{A}$r$\mathbb{B}$o$\mathbb{C}$p$\mathbb{D}$, where $\mathbb{A}$ is the distribution center ID from which the information was extracted, $\mathbb{B}$ is the number of order lines (instance size), $\mathbb{C}$ represents the number of orders, and $\mathbb{D}$ the number of products.

In the next section, we analyze the impact of the parameter γ in the efficiency of the proposed IGPSB.

5.2. *The impact of the affinity value γ*

The γ value, which represents the maximum affinity (or dissimilarity) allowed among product categories assigned to a given rack, is a grouping criterion that may reduce significantly the number of required racks resulting in a potential improvement in picking costs since it decreases the number of places to be visited by the pickers.

As a preliminary analysis to determine its impact in the solution, we evaluate several values of γ. We set these values to $\gamma = 0.0, 0.10, 0.20, 0.30$, where a value of 0.0 corresponds to allow *only identical item categories* in each rack; a value near 1.0, indicates that *items assigned to racks may differ as much as categories* exist. For these tests, we use the *calibration set* and solve only the initial phase of Algorithm 1 (lines 1–2) as this parameter is only used in the MILP for the storage location assignment (SLA) subproblem of the `getInitialSolution` procedure. This MILP (31)-(33) is solved using the CPLEX solver considering a maximum optimization time of 600 s. The final results are presented in Fig. 5-7.

Fig. 5 shows the picking cost per each instance and each affinity value. We can observe that, when only identical product categories can be assigned in a same rack ($\gamma = 0.0$), the picking cost increases, and it is more notorious for larger instances. The reason of this is because instances consider a list of 30 different products for small sizes and 50 products for large instances, and most of these products belong to a different category (only some of them belong to the same category). Then, it is required to use more racks in order to locate all the items in a specific storage location as it is not allowed to place other product categories in a same rack.

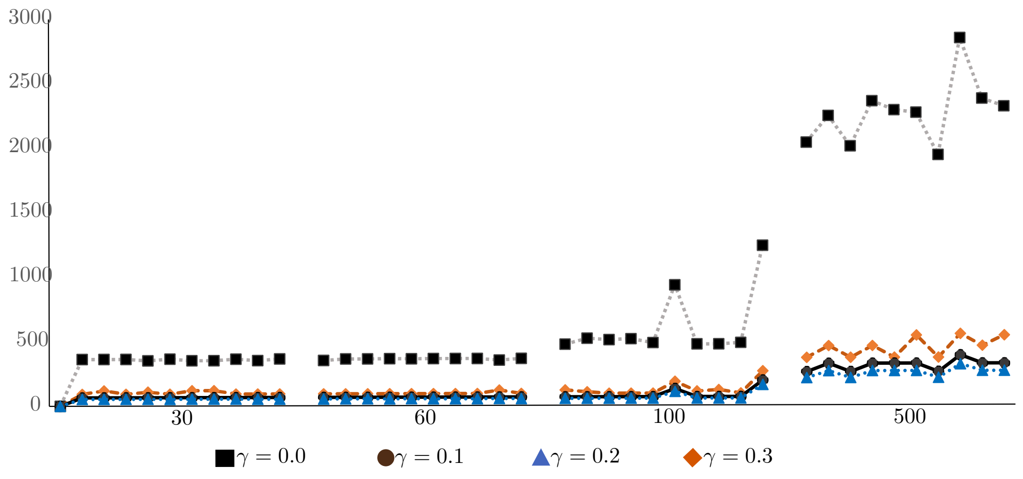


Fig. 5.: Picking cost per affinity level

In Fig. 6, we can observe that the number of racks used to assign items is greater when $\gamma = 0$ (only allowed identical product categories are allowed). On the other hand, when γ increases, the number of required storage locations decreases and the picking costs decrease significantly as this activity requires less number of racks to visit during the routing.

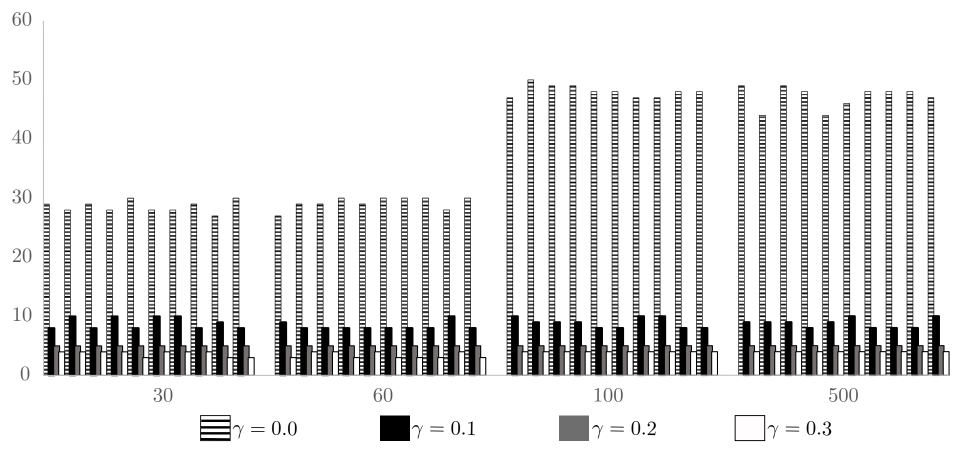


Fig. 6.: Number of racks required per each affinity level

Finally, in Fig. 7, the proportion of computational time required for the constructive heuristic to find the best feasible solution is reported. It can be observed that the most time-consuming option (in all the cases) is presented when $\gamma = 0.10$. Comparing the results shown in Fig. 5-7, $\gamma = 0.20$ outperforms the remaining options as it obtained a better trade-off between picking costs and computing times. Furthermore, the similarity between products located at the same rack is not affected more than the necessary.

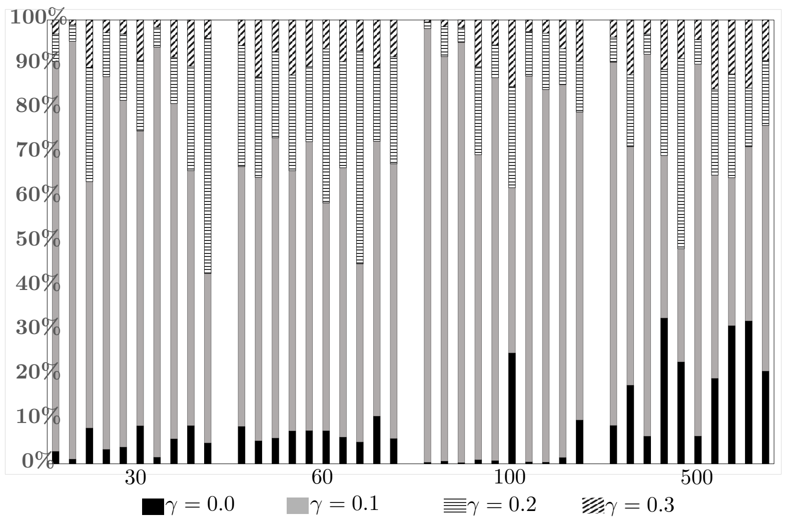


Fig. 7.: Proportion of computing times required using different affinity levels

We conclude that by allowing flexibility when placing products on the racks, order picking costs can be reduced considerably, and a better use of racks space is obtained without modifying significantly the affinity between products assigned to them. Therefore, we propose to use a value of $\gamma = 0.2$ for the remaining experiments.

5.3. *IGPSB performance*

This section is devoted to assess the performance of the proposed solution method. First, we compare the solutions obtained by the IGPSB with respect to those found by solving the MILP formulation (1)-(30) using a general-purpose solver and with respect to the initial feasible solution (improved by the LS) constructed as an input for the IGPSB. Second, since it has been observed that the MILP of the SL assignment subproblem (31)-(33) does not perform efficiently when the number of products is greater than 300, a simple but fast heuristic method is proposed to replace the corresponding MILP (especially for instances in which it does not perform well) conducting several tests to compare both alternatives in the IGPSB.

Calibration tests. In order to better exploit the efficiency of the algorithm, we performed two parameter calibrations required for the IGPSB: the percentage of elements to be *destroyed* (`%Dest`) from the solution and the value of the parameter β, which is used to compute the threshold of acceptance of poorer quality solutions. For these experiments, we use the corresponding set of calibration instances. Final results indicate that better solutions are obtained when `%Dest` $= 30\%$ and $\beta = 0.3$ (more details are available on the repository provided at the beginning of this section).

For all the tests of this section, we consider the parameter setting from the previous section, i.e., a maximum affinity value of $\gamma = 0.20$; the proportion of elements to be removed from a solution of %Dest $= 30$, and a $\beta = 0.3$. The stopping criteria for the IGPSB are set to $\texttt{iterIG}_{\text{max}} = 100$ iterations and $\texttt{MaxOptTime} = 3600$ s. We use ILOG Concert Technology API (CPLEX 20.1) with 3600 s. as time limit to solve the formulation proposed for the problem; and a time limit of 600 s. to solve the SL assignment subproblem to obtain the initial feasible solution. The experiments were conducted on the *Performance set* of instances, considering four repetitions of the IGPSB for each of them. We report the best results of each solution method and the best solution values obtained by the IGPSB.

5.3.1. *Comparison among solution methods*

We now evaluate the results obtained by solving the proposed formulation (1)-(30) as well as the initial and the best solutions found after solving the first and second phases of the IGPSB, respectively. This comparison is shown in Tables 4-5 (for pseudo-real and real-world instances, respectively), where the first column displays the instance name; the next block of three columns (`A - MILP`), corresponds to the costs of the best solution found, the relative optimality gap (%), and the running time (seconds) required by CPLEX to solve the proposed formulation; in block (`B - IG-Ph1`), we show the solution costs and running times required to obtain the (improved) initial feasible solution constructed in the first phase of the IGPSB; on the other hand, column block (`C - IG-Ph2`), provides the solution costs, running times, and the final number of pickers $|W_f|$ needed to perform the routes at the end of the second phase of the IGPSB. Finally, the last column block (`Improvement`) shows the percentage of improvement between each pair of solution methods (`M1-M2`), where a negative value corresponds to an improvement obtained by method `M1` with respect to `M2`.

It can be observed in Table 4 that CPLEX (`A - MILP`) was able to find the optimal solution in 15 out of 30 instances with 30-100 order lines. Feasible instances with a MIP-Gap greater than zero, were re-optimized considering a time limit of 14400s in an attempt to find a better near-optimum solution. It can be noticed that, for the smallest instances (30 order lines), CPLEX was able to find an optimal solution within 4 hours. However, it started to become difficult to solve instance sizes greater or equal than 60 order lines. In fact, for the largest ones no feasible solution was found. On the other hand, the solution algorithm `B` (first phase of the IGPSB), obtained good initial feasible solutions efficiently for the pseudo-real instances requiring at most 604s for the largest ones (most of the computation time was consumed by the SLA subproblem); while the solution method `C` (second IGPSB phase) improved most of those initial solutions from `B`, reporting improvements from 6.9% up to 58.1% in approximately 1h of optimization, outperforming the previous alternatives in terms of solution quality. Considering that the company is interested in knowing the smallest number of pickers required to perform the routes to complete customer orders, we can conclude that for small instances where the number of available pickers is equal to the number of customer orders ($|W| = |O| = 3, 5$), it is needed at most 2 pickers; whereas, for large instances, where $|W| = |O| = 150, 250$, only required from 4 to 8 pickers.

Table 4.: Performance comparison among solution methods on pseudo-real instances

Instance	A - MILP			B - IG-Ph1		C - IG-Ph2			Improvement		
	Sol	**Gap (%)**	**Time**	**Sol**	**Time**	**Sol**	**Time**	$\lvert W_f \rvert$	**(B-A)**	**(C-A)**	**(C-B)**
Inst1r30o3p20	52.5	0.0	1005.5	52.5	1.0	52.5	33.6	1	0.0%	0.0%	0.0%
Inst2r30o3p20	52.5	0.0	753.4	52.5	0.8	52.5	39.8	1	0.0%	0.0%	0.0%
Inst3r30o3p20	63.9	0.0	59.1	68.7	1.1	63.9	41.1	1	7.4%	0.0%	-6.9%
Inst4r30o3p20	52.5	0.0	276.4	52.5	0.9	52.5	41.3	1	0.0%	0.0%	0.0%
Inst5r30o3p20	52.5	0.0	404.5	52.5	1.1	52.5	40.2	1	0.0%	0.0%	0.0%
Inst1r30o5p20	63.9	0.0	**11010.7**	68.7	1.0	63.9	42.8	1	7.4%	0.0%	-6.9%
Inst2r30o5p20	52.9	0.0	**4570.8**	52.5	0.8	52.5	41.5	1	0.0%	0.0%	0.0%
Inst3r30o5p20	63.9	0.0	**4198.0**	68.7	1.1	63.9	43.7	1	7.4%	0.0%	-6.9%
Inst4r30o5p20	63.9	0.0	**14398.5**	68.7	1.2	63.9	42.9	1	7.4%	0.0%	-6.9%
Inst5r30o5p20	63.9	0.0	**8023.7**	68.7	0.9	63.9	46.6	1	7.4%	0.0%	-6.9%
Inst1r60o3p30	139.4	62.9	**14399.2**	137.4	2.8	127.9	201.7	2	-1.4%	-8.2%	-6.9%
Inst2r60o3p30	137.4	55.7	**14398.7**	137.4	4.3	127.9	87.9	2	0.0%	-6.9%	-6.9%
Inst3r60o3p30	63.9	0	2049.8	68.7	3.7	63.9	48.8	1	7.4%	0.0%	-6.9%
Inst4r60o3p30	63.9	0	365.7	68.7	2.9	63.9	50.1	1	7.4%	0.0%	-6.9%
Inst5r60o3p30	63.9	0	1404.8	68.7	4.9	63.9	44.5	1	7.4%	0.0%	-6.9%
Inst1r60o5p30	127.9	54.2	**14400.9**	137.4	4.2	113.6	323.0	2	7.4%	-11.2%	-17.3%
Inst2r60o5p30	593.6	93.6	**14400.7**	137.4	5.5	102.7	280.1	2	-76.9%	-82.7%	-25.3%
Inst3r60o5p30	63.9	0	818.9	68.7	4.2	63.9	41.7	1	7.4%	0.0%	-6.9%
Inst4r60o5p30	606.3	91.7	**14400.8**	137.4	2.4	116.5	288.5	2	-77.3%	-80.8%	-15.2%
Inst5r60o5p30	63.9	0	1014	68.7	2.5	63.9	37.2	1	7.4%	0.0%	-6.9%
Inst1r100o3p50	127.9	26.4	**14395.5**	137.4	22.0	127.9	114.5	2	7.4%	0.0%	-6.9%
Inst2r100o3p50	639.3	86.6	**14394.3**	137.4	21.5	127.9	92.1	2	-78.5%	-80.0%	-6.9%
Inst3r100o3p50	137.4	45.7	**14395.3**	137.4	18.9	127.9	86.1	2	0.0%	-6.9%	-6.9%
Inst4r100o3p50	388.9	81.5	**14394.9**	137.4	8.5	127.9	87.9	2	-64.7%	-67.1%	-6.9%
Inst5r100o3p50	127.9	25.8	**14395.6**	137.4	9.5	127.9	81.9	2	7.4%	0.0%	-6.9%
Inst1r100o5p50	748.7	92.3	**14399.3**	137.4	21.1	127.9	331.9	2	-81.7%	-82.9%	-6.9%
Inst2r100o5p50	915.6	93.6	**14398.7**	137.4	28.3	127.9	277.7	2	-84.5%	-85.6%	-6.9%
Inst3r100o5p50	—	—	**14400.1**	137.4	8.8	127.9	80.9	2	—	—	-6.9%
Inst4r100o5p50	—	—	**14400.0**	137.4	55.0	127.9	198.0	2	—	—	-6.9%
Inst5r100o5p50	—	—	**14400.0**	137.4	23.1	127.9	159.1	2	—	—	-6.9%
Inst1r1000o150p100	—	—	—	169.7	601.3	127.9	3648.9	2	—	—	-24.6%
Inst2r1000o150p100	—	—	—	137.4	601.2	127.9	3647.7	2	—	—	-6.9%
Inst3r1000o150p100	—	—	—	137.4	601.2	127.9	3718.2	2	—	—	-6.9%
Inst4r1000o150p100	—	—	—	169.7	601.5	127.9	3716.9	2	—	—	-24.6%
Inst5r1000o150p100	—	—	—	305.0	601.8	127.9	3762.9	2	—	—	-58.1%
Inst1r1000o250p100	—	—	—	169.7	601.7	127.9	3948.6	2	—	—	-24.6%
Inst2r1000o250p100	—	—	—	150.8	602.5	127.9	4013.1	2	—	—	-15.2%
Inst3r1000o250p100	—	—	—	137.4	601.7	127.9	3638.7	2	—	—	-6.9%
Inst4r1000o250p100	—	—	—	137.4	601.1	127.9	3689.2	2	—	—	-6.9%
Inst5r1000o250p100	—	—	—						—	—	
Inst1r3000o150p100	—	—	—	274.8	601.4	255.8	3668.5	4	—	—	-6.9%
Inst2r3000o150p100	—	—	—	385.1	601.4	301.4	3660.7	4	—	—	-21.7%
Inst3r3000o150p100	—	—	—	274.8	601.9	255.8	3928.9	4	—	—	-6.9%
Inst4r3000o150p100	—	—	—	457.7	602.4	319.7	3893.4	5	—	—	-30.1%
Inst5r3000o150p100	—	—	—	339.4	602.5	255.8	3922.7	4	—	—	-24.6%
Inst1r3000o250p100	—	—	—	274.8	602.8	255.8	3807.7	4	—	—	-6.9%
Inst2r3000o250p100	—	—	—	274.8	601.8	255.8	3847.8	4	—	—	-6.9%
Inst3r3000o250p100	—	—	—	274.8	463.0	255.8	3711.2	4	—	—	-6.9%
Inst4r3000o250p100	—	—	—	339.4	601.2	255.8	3655.2	4	—	—	-24.6%
Inst5r3000o250p100	—	—	—	343.5	229.1	319.7	3753.4	5	—	—	-6.9%
Inst1r5000o150p100	—	—	—	464.2	568.5	447.6	3819.5	7	—	—	-3.6%
Inst2r5000o150p100	—	—	—	480.8	298.1	447.6	3811.6	7	—	—	-6.9%
Inst3r5000o150p100	—	—	—	640.8	602.6	447.6	4032.3	7	—	—	-30.1%
Inst4r5000o150p100	—	—	—	480.8	603.0	447.6	3913.8	7	—	—	-6.9%
Inst5r5000o150p100	—	—	—	480.8	603.5	447.6	4064.6	7	—	—	-6.9%
Inst1r5000o250p100	—	—	—	754.0	603.8	527.6	4305.6	7	—	—	-30.0%
Inst2r5000o250p100	—	—	—	412.1	602.0	383.6	3639.7	6	—	—	-6.9%
Inst3r5000o250p100	—	—	—	480.8	506.8	447.6	3733.0	7	—	—	-6.9%
Inst4r5000o250p100	—	—	—	594.0	601.5	447.6	3859.2	7	—	—	-24.6%
Inst5r5000o250p100	—	—	—	549.5	601.4	511.5	3775.5	8	—	—	-6.9%

Table 5 shows a similar description but now considering the Subset B of instances (real-world). For this set, CPLEX was not able to find the optimal solution, even though the number of order lines is around 100 and customer orders does not exceed 44. An important characteristic of this set is that the number of products is similar than the number order lines, which is an important difference with respect to the pseudo-real instances, having an important effect on the efficiency of the proposed algorithms. In addition, an important observation during the tests was that for instance sizes larger than 5000 order lines (and a number of products greater than 2000), no feasible solutions could be found for either **A** or the MILP subproblem solved in **B**. This was because CPLEX was not able to load those instances. Nevertheless, the IGPSB avoids that issue by using the SLA heuristic alternative to construct the initial solution when the MILP subproblem (31)-(33) is not enough (see Section 4.1).

Table 5.: Performance comparison among solution methods on real-world instances

Instance	**A - MILP**			**B - IG-Ph1**		**C - IG-Ph2**		**Improvement**		
	Sol	**Gap (%)**	**Time**	**Sol**	**Time**	**Sol**	**Time**	**(B-A)**	**(C-A)**	**(C-B)**
Int23r101o5p98	140.8	70.4	3603.1	107.7	601.0	63.9	647.0	-23.5%	-54.6%	-40.6%
Int85r107o1p107	210.8	75.6	3600.3	80.1	601.1	63.9	642.0	-62.0%	-69.7%	-20.2%
Inst53r110o8p107	80.1	28.4	14400.3	261.8	601.2	63.9	638.9	226.7%	-20.2%	-75.6
Inst58r111o7p106	137.5	70.3	3607.0	201.5	601.4	107.7	863.9	46.5%	-21.7%	-46.5%
Inst98r111o9p106	289.3	85.9	3613.7	91.5	601.1	63.9	637.7	-68.4%	-77.9%	-30.2%
Inst22r362o30p325	—	—	—	119.1	606.9	80.1	665.4	—	—	-32.8%
Inst22r457o44p376	—	—	—	815.5	708.5	702.9	1623.1	—	—	-13.8%
Int121r463o36p411	—	—	—	1033.0	683.1	810.5	2229.3	—	—	-21.5%
Inst93r467o32p414	—	—	—	125.8	610.7	125.8	668.6	—	—	0.0%
Inst154r477o33p420	—	—	—	782.3	630.3	568.0	2017.5	—	—	-27.4%
Inst134r5064o384p2923	—	—	—	343.4	173.1	319.7	3711.9	—	—	-6.9%
Inst2r5528o399p3128	—	—	—	343.45	33.8	319.7	3277.5	—	—	-6.9%
Inst22r7643o654p3944	—	—	—	593.95	261.2	447.6	3754.3	—	—	-24.6%
Inst108r8197o596p4252	—	—	—	678.8	340.0	511.5	4118.2	—	—	-24.6%

5.4. *Performance under different demand scenarios*

In this section, we conducted the following tests to compare the performance of the exact methods applied by CPLEX when solving the proposed MILP versus the IG-PSB performance. For each size (30 and 60 order lines), we generated 10 pseudo-real instances, half of them with 3 customer orders and the remaining ones with 5 orders. Two demand scenarios were considered. The first scenario considers a positively skewed demand distribution, with 80% of products having a demand between 1 and 3 units, and the remaining 20% of products, between 6 and 10 units. The second scenario corresponds to a negatively skewed demand distribution, with 80% of products having a demand between 6 and 10 units, and 20% of products with a demand between 1 and 3 units. We set a time limit of 3600 s for both solution methods. The solution quality for the IGPSB corresponds to the best solution found of four repetitions of the algorithm and, its computing times correspond to the average value. Notice that a third scenario (the *symmetrical* distribution) corresponds to the case of the pseudo-real instances shown in the previous sections, where demand is uniformly distributed on the interval $\sim U[1, 3]$. However, this case is not considered for this comparison.

Table 6 shows the results obtained by CPLEX ("MILP" columns) and the IGPSB ("IGPSB" columns). It can be observed that the most easiest scenario corresponds to the subset of instances of size 30 and a positive skewness demand distribution (SMALL DEMAND predominance), since most of them were solved to optimality requiring less computing time for both solution methods. A tendency of increasing complexity can be observed when the number of order lines increases and the demand distribution shifts towards larger demands (LARGE DEMAND predominance). This is most noticeable in CPLEX performance, in which the relative optimality **Gap** was mostly very large (even no solutions were found in several cases) and computing times were consumed to the limit. The opposite case was for the IGPSB, which performed better in both scenarios. Notice that, only for the second instance with $n = 30$, $|O| = 3$, and large demand, the IGPSB obtained a much worse solution than the MILP. For that case, the initial solution was not improved in 160.6 s., this establishes a guideline for exploring other parameter settings to the algorithm or adopting different strategies for improving the solution for specific cases.

Table 6.: Performance of solutions methods considering two different demand scenarios.

		SMALL DEMAND						**LARGE DEMAND**					
		MILP			**IGPSB**			**MILP**			**IGPSB**		
n	$\|O\|$	**Sol**	**Gap**	**Time**	**Sol**	**Time**	**Imp**	**Sol**	**Gap**	**Time**	**Sol**	**Time**	**Imp**
30	3	52.5	0.0	1070.2	52.5	28.9	0.0%	63.9	0.0	334.1	63.9	35.3	0.0%
		52.5	0.0	124.0	52.5	30.2	0.0%	157.6	0.0	4026.3	262.6	160.6	66.6%
		63.9	29.1	3600.1	63.9	30.3	0.0%	63.9	0.0	692.4	63.9	34.2	0.0%
		52.5	0.0	942.0	52.5	28.6	0.0%	104.2	25.3	3600.0	105.0	61.6	0.8%
		105.1	51.2	3600.2	105.0	54.5	0.0%	157.6	59.3	3599.6	157.6	93.6	0.0%
	5	52.5	0.0	156.9	52.5	28.6	0.0%	105.1	46.4	71991.9	105.0	64.2	0.0%
		52.5	19.1	3601.4	52.5	28.3	0.0%	137.4	51.5	3600.8	116.5	74.8	-15.2%
		177.3	78.3	3601.2	52.5	29.3	-70.4%	116.5	52.7	71991.6	116.5	80.6	0.0%
		52.5	0.0	752.2	52.5	28.2	0.0%	295.4	84.2	3601.0	105.0	316.3	-64.4%
		63.9	0.0	1219.1	63.9	29.9	0.0%	672.8	91.9	3600.9	116.5	222.7	-82.7%
Average				**1866.7**		**31.7**	**-7.1%**			**16703.9**		**114.4**	**-9.5%**
60	3	261.0	72.7	3600.3	127.9	112.2	-51.0%	1079.4	94.2	3598.0	127.9	64.7	-88.2%
		63.9	0.0	252.8	127.9	69.2	99.9%	—	—	3600.0	127.9	69.4	—
		63.9	0.0	2324.9	127.9	60.1	99.9%	—	—	3600.0	180.4	99.8	—
		127.9	48.1	3600.2	105.0	115.7	-17.9%	137.39	52.6	3598.2	127.9	71.0	-6.9%
		63.9	0.0	1010.1	116.5	62.4	82.1%	—	—	3600.0	191.8	102.8	—
	5	563.1	91.5	3602.0	127.9	61.6	-77.3%	—	—	3600.0	221.5	235.9	—
		652.9	92.5	3601.3	116.5	156.2	-82.2%	—	—	3600.0	232.9	287.3	—
		544.1	88.8	3601.9	105.0	167.3	-80.7%	—	—	3600.0	191.8	297.6	—
		—	—	3600.3	127.9	242.6	—	—	—	3600.0	232.9	246.2	—
		—	—	3600.3	152.3	291.4	—	—	—	3600.0	221.5	173.5	—
Average				**2879.4**		**133.9**	**-3.4%**			**3599.6**		**164.8**	**-47.5%**

6. Conclusions

This paper addresses an order picking problem, which integrates storage location, batching, and routing decisions. The aim is to find the allocation of products to racks, the grouping of customer orders, and the sequence of routes performed by pickers that minimize the order picking cost. A mixed-integer linear programming model is introduced, and an Iterated Greedy Local Search algorithm (IGPSB) is proposed to solve the problem. Comprehensive computational experiments have been carried out to calibrate the main parameters of the IGPSB and to assess the performance of the proposed solution methods on pseudo-real and real-world instances focused on the basis of a Mexican retail firm. During the performance tests, we have observed that the commercial branch-and-bound method applied directly to the MILP model, used to solve the SLA subproblem when the initial feasible solution is constructed, was able to solve all the pseudo-real instances. However, for the real-world instances, the branch-and-bound method started to present problems, especially for larger sizes where it was not able to load the problem into the solver. Therefore, a simple but fast heuristic method was developed to obtain the initial location assignment of products obtaining initial solutions very quickly. With this new proposal, final results have shown that the IGPSB can obtain feasible solutions in reasonable computing times for instances with up to 8197 order lines, 654 customer orders, and up to 4252 products. On the other hand, the proposed model solved by a commercial state-of-the-art solver can only achieve optimal solutions for a few small size instances, not being able to solve those of larger size. Moreover, we have conducted a comprehensive assessment of the effectiveness of the proposed methods varying customer demand conditions (small and large demand scenarios). The results showed that the solver performs better in scenarios that consider small demands, while the IGPSB consistently outperforms all the scenarios.

Future research lines identified during the development of this work are devoted to strengthening the proposed formulation by developing other valid inequalities or the introduction of cuts during the optimization process to reduce the symmetries in batching and routing decisions. Another research direction is to extend the problem scope by considering the integration of other decisions such as zoning, packaging, or last-mile operations. Also, it may be interesting to study the use of other routing, batching, and storage location policies to make a comparison with the strategies proposed in this work; or use other criteria in the objective function, for example, consider a penalization for due orders or introduce a decision that considers inventory levels in order to know when to replenish product in racks. In terms of robustness, it would be interesting to make a complete study of the proposed algorithms considering other layouts, a heterogeneous fleet of picker carts, or integrating dynamic and stochastic elements.

Acknowledgments

The research of Diana L. Huerta-Muñoz has been supported by the National Council of Humanities, Science and Technology (CONAHCYT) through a postdoctoral fellowship. The research of Roger Z. Ríos-Mercado has been partially funded by CONAHCYT (grant FC-2016-2/194), and Universidad Autónoma de Nuevo León (UANL, grants PAICYT CE1837-21 and 241-CE-2022). The research of Jesús F. López-Pérez has been partially supported by UANL. This support is gratefully acknowledged.

CRediT authorship contribution statement

Diana L. Huerta-Muñoz: Conceptualization, Validation, Formal analysis, Investigation, Writing - Original Draft, Writing - Review & Editing, Data Curation, Methodology, Software, and Visualization. **Roger Z. Ríos-Mercado:** Conceptualization, Validation, Formal analysis, Investigation, Writing - Review & Editing, Supervision, Resources, and Project administration. **Jesús F. López-Pérez:** Conceptualization, Validation, Formal analysis, Investigation, Writing - Review & Editing, Supervision, and Data Curation.

Data availability statement

The authors confirm that the data supporting the findings of this study are available within the article [and/or] its supplementary materials.

Declaratation of interest

None.

References

Albareda-Sambola, M., A. Alonso-Ayuso, E. Molina, and C. S. De Blas (2009). Variable neighborhood search for order batching warehouse. *Asia-Pacific Journal of Operational Research 26*(05), 655–683.

Braekers, K., K. Ramaekers, and I. Van Nieuwenhuyse (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering 99*, 300–313.

Cao, Z., L. Zhou, C. Lin, and M. Zhou (2023). Solving an order batching, picker assignment, batch sequencing and picker routing problem via information integration. *Journal of Industrial Information Integration 31*, 100414.

Cardona, L. F., L. Rivera, and H. J. Martnez (2012). Analytical study of the fishbone warehouse layout. *International Journal of Logistics Research and Applications 15*(6), 365–388.

Celik, M., C. Archetti, and H. Sural (2022). Inventory routing in a warehouse: The storage replenishment routing problem. *European Journal of Operational Research 301*(3), 1117–1132.

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

Chen, C.-M., Y. Gong, R. B. M. de Koster, and J. A. E. E. van Nunen (2010). A flexible evaluative framework for order picking systems. *Production and Operations Management 19*(1), 70–82.

Chen, W., Y. Zhang, and Y. Zhou (2022). Integrated scheduling of zone picking and vehicle routing problem with time windows in the front warehouse mode. *Computers & Industrial Engineering 163*, 107823.

Clarke, G. and J. W. Wright (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research 12*(04), 568–581.

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

Dukic, G. and T. Opetuk (2012). Warehouse layouts. In R. Manzini (Ed.), *Warehousing in the Global Supply Chain: Advanced Models, Tools and Applications for Storage Systems*, Chapter 3, pp. 55–69. London, UK: Springer.

Floyd, R. W. (1962). Algorithm 97: shortest path. *Communications of the ACM 5*(6), 345.

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

Gutin, G. and A. P. Punnen (2007). *The Traveling Salesman Problem and Its Variations*. New York: Springer.

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

Henn, S., S. Koch, and G. Wäscher (2012). Order batching in order picking warehouses: A survey of solution approaches. In R. Manzini (Ed.), *Warehousing in the Global Supply Chain*, Chapter 6, pp. 105–137. London, UK: Springer.

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

Hsieh, L.-F. and Y.-C. Huang (2011). New batch construction heuristics to optimise the performance of order picking systems. *International Journal of Production Economics 131*(2), 618–630.

Kofler, M., A. Beham, S. Wagner, and M. Affenzeller (2014). Affinity based slotting in warehouses with dynamic order patterns. In R. Klempous, J. Nikodem, W. Jacak, and Z. Chaczko (Eds.), *Advanced Methods and Applications in Computational Intelligence*, Volume 6 of *Topics in Intelligent Engineering and Informatics*, Chapter 7, pp. 123–143. Cham, Switzerland: Springer.

Kübler, P., C. H. Glock, and T. Bauernhansl (2020). A new iterative method for solving the joint dynamic storage location assignment, order batching and picker routing problem in manual picker-to-parts warehouses. *Computers & Industrial Engineering 147*, 106645.

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

Lodi, A., S. Martello, and D. Vigo (2002). Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics 123*(1), 379–396.

Loiola, E. M., N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido (2007). A survey for the quadratic assignment problem. *European Journal of Operational Research 176*(2), 657–690.

Lourenço, H. R., O. C. Martin, and T. Stützle (2003). Iterated local search. In F. Glover and G. A. Kochenberger (Eds.), *Handbook of Metaheuristics*, Volume 57 of *International Series in Operations Research & Management Science*, Chapter 11, pp. 321–353. Boston: Springer.

Masae, M., C. H. Glock, and E. H. Grosse (2020). Order picker routing in warehouses: A systematic literature review. *International Journal of Production Economics 224*, 107564.

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

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

Rojas Reyes, J. J., E. L. Solano-Charris, and J. R. Montoya-Torres (2019). The storage location assignment problem: A literature review. *International Journal of Industrial Engineering Computations 10*(2), 199–224.

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

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

Theys, C., O. Brysy, W. Dullaert, and B. Raa (2010). Using a TSP heuristic for routing order pickers in warehouses. *European Journal of Operational Research 200*(3), 755–763.

Tompkins, J. A., J. A. White, Y. A. Bozer, and J. M. A. Tanchoco (2010). *Facilities Planning.* New York: Wiley.

van Gils, T., A. Caris, K. Ramaekers, and K. Braekers (2019). Formulating and solving the integrated batching, routing, and picker scheduling problem in a real-life spare parts warehouse. *European Journal of Operational Research 277*(3), 814–830.

van Gils, T., A. Caris, K. Ramaekers, K. Braekers, and R. B. M. de Koster (2019). Designing efficient order picking systems: The effect of real-life features on the relationship among planning problems. *Transportation Research Part E: Logistics and Transportation Review 125*, 47–73.

van Gils, T., K. Ramaekers, K. Braekers, B. Depaire, and A. Caris (2018). Increasing order picking efficiency by integrating storage, batching, zone picking, and routing policy decisions. *International Journal of Production Economics 197*, 243–261.

van Gils, T., K. Ramaekers, A. Caris, and R. B. M. de Koster (2018). Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. *European Journal of Operational Research 267*(1), 1–15.

Wagner, S. and L. Mönch (2023). A variable neighborhood search approach to solve the order batching problem with heterogeneous pick devices. *European Journal of Operational Research 304*(2), 461–475.

Wang, M., R.-Q. Zhang, and K. Fan (2020). Improving order-picking operation through efficient storage location assignment: A new approach. *Computers & Industrial Engineering 139*, 106186.

Wang, Z., W. Xu, X. Hu, and Y. Wang (2022). Inventory allocation to robotic mobile-rack and picker-to-part warehouses at minimum order-splitting and replenishment costs. *Annals of Operations Research 316*(1), 467–491.

Won, J. and S. Olafsson (2005). Joint order batching and order picking in warehouse operations. *International Journal of Production Research 43*(7), 1427–1442.

Zhen, L., J. Wu, H. Li, Z. Tan, and Y. Yuan (2023). Scheduling multiple types of equipment in an automated warehouse. *Annals of Operations Research 322*(2), 1119–1141.

7. Appendix. Calibration tests and comparison of SLA algorithms

In order to better exploit the efficiency of the algorithm, we have performed two required parameter calibrations for the IGPSB: the percentage of elements to be *destroyed* (`%Dest`) from the solution and the value of the parameter β, which is used to compute the threshold of acceptance of poorer quality solutions. For these experiments, we use the corresponding set of calibration instances and fix the affinity value to $\gamma = 0.20$ (as it obtained better results in the preliminary tests), the time limit to solve the MILP used to construct an initial feasible SLA is set to 600 s, and the stopping criteria for the IGPSB are fix to $\texttt{iterIG}_{\max} = 100$ iterations and `MaxOptTime` = 3600 s.

7.1. *Calibration of percentage of solution destruction parameter*

The first parameter to calibrate is `%Dest`, the number of elements to remove from a solution during the `Destruction` step (line 5, Algorithm 1). We consider three factor levels, `%Dest` = 30, 50, and 70, which mean that, for a given solution s, it is required to remove (randomly) 30/50/70% of the total number of selected racks, which later will be reassigned (greedily) in the same proportion during the `reConstruction` procedure. For these specific experiments, we also fix the parameter $\beta = 0$, which corresponds to the option of not accepting poor-quality solutions during the IGPSB. Since the `Destruction` procedure applies a random selection of solution elements to remove, we consider 4 repetitions of the IGPSB for each tested instance.

Table 7 shows the results obtained per each `%Dest` and instance size. The first column is assigned to the instance size; in the second one, the `%Dest` values to be evaluated are shown; and, in column blocks RPD and Time(s), we provide the minimum (Min), the average (Avg), and the maximum (Max) ARP/Time obtained per each group size of the calibration set. It is observed for small instances ($r = 30, 60$) that a local optimum was found, and the perturbations applied to the solutions were enough to find a good result in few seconds providing no difference among the `%Dest` options in terms of solution quality; however, in terms of computing times, removing a 30% of the solution requires less time than removing 50 or 70%. On the other hand, for large instances ($r = 100, 500$), it is easier to identify the options that obtain the best results providing more information about the algorithm performance when the value of `%Dest` varies. We can observe that the best RPD values and computing times are obtained when the percentage of the elements to remove is 30%, obtaining 1.30% of ARPD in 843.7s on average (the maximum RPD is also small for this case). Therefore, the value of `%Dest` = 30 is definitely the percentage to be destroyed of a given solution s in the IGPSB as it outperforms the other options.

7.1.1. Calibration of β

The second parameter to calibrate corresponds to β, which is a value in the [0,1] range used to calculate the threshold of acceptance of poor-quality solutions during the IGPSB (see Algorithm 4). The smaller the β value, the tighter the acceptance threshold and fewer solutions with worse objective value will be accepted. A value near 1 allows to accept any solution that falls within the current gap computed between the best solution found so far and the last (average) solution obtained in the current iteration. For these tests, we set this parameter to $\beta = 0.0, 0.1, 0.2$ and 0.3. Since it is important that the algorithm has a mechanism for escaping from local optima, the

Table 7.: Average RPD and computing times by removing a % of the solution in the IGPSB

Instance	%Dest	RPD			Time(s)		
		Min	Avg	Max	Min	Avg	Max
r30	**30**	0.00%	0.00%	0.00%	200.5	272.1	363.6
r60		0.00%	0.00%	0.00%	205.0	276.9	369.9
r100		0.00%	0.02%	0.08%	339.7	441.0	533.4
r500		0.00%	5.18%	16.60%	1855.4	2384.6	2940.7
Avg		**0.00%**	**1.30%**	**4.17%**	**650.1**	**843.7**	**1051.9**
r30	**50**	0.00%	0.00%	0.00%	323.9	444.5	646.1
r60		0.00%	0.00%	0.00%	341.0	428.3	554.2
r100		0.00%	1.11%	4.35%	419.5	551.4	676.5
r500		4.65%	16.42%	35.24%	2364.5	2977.8	3399.2
Avg		1.16%	4.38%	9.90%	862.2	1100.5	1319.0
r30	**70**	0.00%	0.00%	0.00%	363.6	646.1	511.9
r60		0.00%	0.00%	0.00%	369.9	554.2	610.7
r100		0.08%	4.35%	0.08%	533.4	676.5	533.4
r500		16.6%	35.24%	23.63%	2940.7	3399.2	3432.7
Avg		4.17%	9.90%	5.93%	1051.9	1319.0	1272.2

parameter $\beta = 0.0$ will be used only for comparative terms on the degradation of the quality of the solution when considering greater values of this parameter. The aim is to determine the $\beta > 0.0$ that obtains the best trade-off between the average RPD and the average computing times. We consider all the parameters mentioned in the previous tests including `%Dest` $= 30$. Four repetitions of the IGPSB are carried out for each tested instance.

The description of Table 8 is similar to that of Table 7, but now the comparison is with respect to β. According to the results shown in Table 8, there is a clear option that provides better results on average, which corresponds to $\beta = 0.3$. We can observe that, although the average RPD has been slightly better when $\beta = 0.0$, this option does not allow to escape from local optimal eliminating any possibility of improving the solution. Furthermore, the number of best solutions found, and the average computing times are better when $\beta = 0.3$, the latter giving the opportunity to further explore the solution space (if the stopping criteria allow it) in order to find better solutions at the end of the optimization. Therefore, $\beta = 0.3$ is the value that provides the best trade-off among the evaluated options.

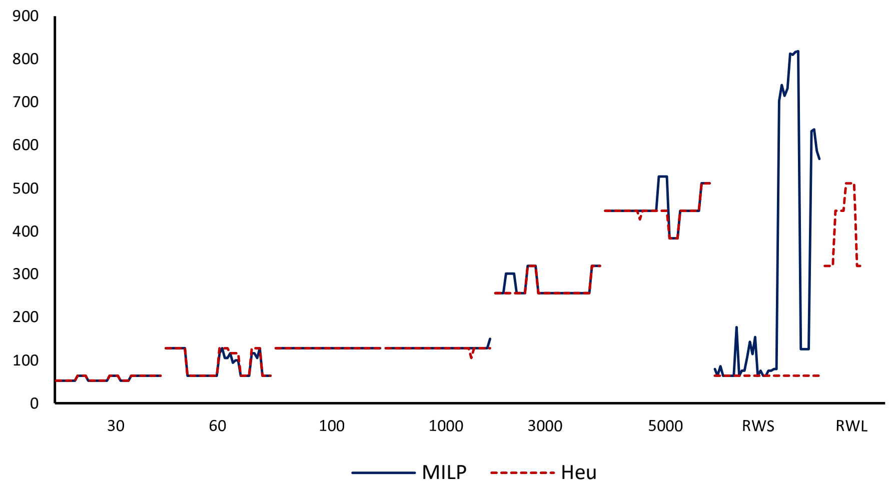
Table 8.: RPD and computing times considering different β values in the IGPSB

Instance	β	#Best	RPD			Time(s)		
			Min	Avg	Max	Min	Avg	Max
r30	**0.0**	10/10	0.00%	0.00%	0.00%	200.50	272.09	363.64
r60		10/10	0.00%	0.00%	0.00%	204.96	276.92	369.85
r100		10/10	0.00%	0.00%	0.00%	339.67	440.97	533.36
r500		10/10	0.00%	4.32%	10.49%	1855.42	2384.62	2940.68
	Avg		**0.0%**	**1.1%**	**2.6%**	**650.14**	**843.65**	**1051.88**
r30	**0.1**	10/10	0.00%	0.00%	0.00%	76.80	81.15	85.98
r60		10/10	0.00%	0.00%	0.00%	82.07	87.28	92.72
r100		10/10	0.00%	1.11%	4.35%	271.52	310.00	385.08
r500		8/10	4.35%	11.54%	24.30%	1564.97	1659.98	1764.41
	Avg		1.1%	3.2%	7.2%	498.84	534.60	582.05
r30	**0.2**	10/10	0.00%	0.00%	0.00%	78.92	83.86	88.82
r60		10/10	0.00%	0.00%	0.00%	81.23	88.47	96.56
r100		10/10	0.00%	1.11%	4.35%	265.02	309.73	376.41
r500		8/10	4.35%	12.95%	27.60%	1594.46	1724.32	1910.12
	Avg		1.1%	3.5%	8.0%	504.91	551.59	617.98
r30	**0.3**	10/10	0.00%	0.00%	0.00%	78.72	83.21	88.69
r60		10/10	0.00%	0.00%	0.00%	81.08	88.59	97.04
r100		10/10	0.00%	1.11%	4.35%	269.22	309.13	371.56
r500		9/10	1.20%	9.66%	23.04%	1575.94	1668.64	1761.99
	Avg		**0.3%**	**2.7%**	**6.8%**	**501.24**	**537.39**	**579.82**

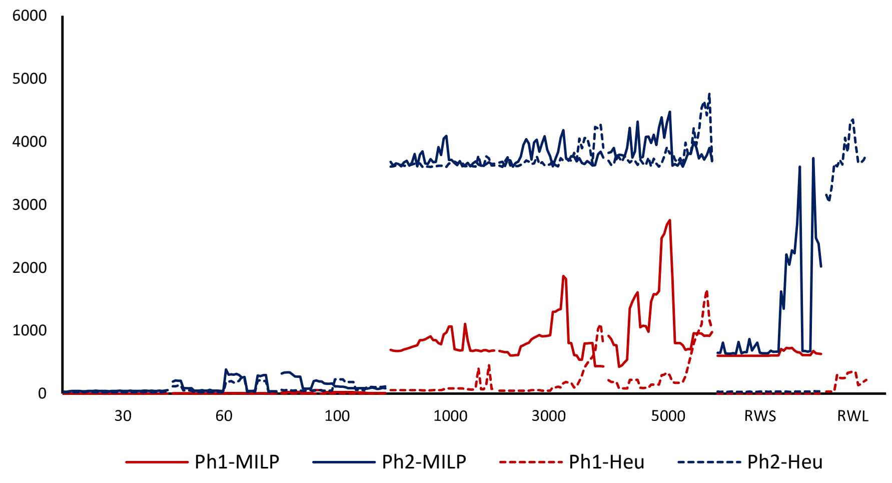
7.2. *SLA MILP vs SLA Heuristic*

In real-world instances, when the instance size is large, it is difficult to introduce the problem into CPLEX to be solved. Therefore, no solution for the SL assignment subproblem can be created, showing a limitation to perform the second phase of the IGPSB. As an alternative to address this issue, we have proposed a simple heuristic to create an initial assignment of products into racks and thus continue with the batching and routing decisions to generate a first initial feasible solution for the IGPSB. The proposed heuristic, starts by sorting the set of products and available racks according to their affinity and their closeness to the depot, respectively. Then, in a sequential order, the first rack l is *open* and the first product p is assigned to it if there exists available space to locate (at least) a proportion of its total demand and its affinity value, with respect to the first product $\bar{p}$ assigned to l, is less than γ; otherwise, the current rack is no longer considered and another one must be open to locate the products that are not yet assigned. Once the total units of products p are place into racks, the product is removed from the non-assigned set. Finally, a feasible assignment is provided if all products were located in a given storage location.

The heuristic was embedded in the first phase of the IGPSB replacing the MILP that solves the initial SL assignment. Then, the IGPSB was performed (four repetitions per each instance) and results are briefly summarized in Figures 8a-8b, where the solution costs and computing times provided by the IGPSB, considering the MILP and the SLA heuristic, are compared. It is observed in Figure 8a that, for the pseudo-real instances (30-5000 order lines), both procedures perform similar; in this case, the SLA heuristic (dashed red lines) obtained improvements of up to 18% in five of the large instances of this set. On the other hand, for small and large real-world instances (RWS, RWL, respectively), significant differences are found (up to 91% of improvement); in fact, as it has been pointed, the MILP for the SLA subproblem has complications in carrying out the load of RWL instances and no feasible solutions can be found to proceed; so, the proposed heuristic can solve this problem. In terms of computing times, Figure 8b provides the required time for each phase of the IGPSB considering both alternatives. For the first IGPSB phase (obtaining an improved initial solution), the SLA heuristic (dashed red lines) outperforms the MILP alternative (solid red lines) in most of the cases, especially for larger instances. For the second phase, the time required for the IGPSB is similar in both cases on pseudo-real instances; however, the most noticeable results, in which the performance of the heuristic is superior, correspond to those provided for real-world instances. Both alternatives, solving directly the MILP with the branch and bound of the solver and the heuristic implemented for the SLA subproblem, seem to perform similar for small and medium instance sizes. We suggest to solve directly the MILP with branch and bound only for instances with less than 5000 order lines, 250 customer orders, and 100 available products as we have observed that, when the number of products increases, the MILP is harder to solve by a general-purpose solver (even if the number of order lines is small). For the remaining cases, the proposed SLA heuristic is used to obtain the initial SLA for the initial solution of the PSB.



(a) Final IGPSB solution costs



(b) Total computing times for phase 1 and 2 of the IGPSB

Fig. 8.: Comparison of final solution costs and computing times using the SLA MILP vs the pure heuristic in the IGPSB