

Minimizing Repetitive Picks in Multiwave Distribution: A Case Study at Lojas Renner

Marçal Quintino Ferreira

Instituto Tecnológico de Aeronáutica (ITA); Universidade Federal de São Paulo (UNIFESP)
São José dos Campos, SP, Brazil
marcal.ferreira@unifesp.br

Bruno Gabriel

Universidade Federal de São Paulo (UNIFESP)
São José dos Campos, SP, Brazil
bruno.sant@unifesp.br

Maria Clara

Universidade Federal de São Paulo (UNIFESP)
São José dos Campos, SP, Brazil
clara.couto@unifesp.br

Renato Milhomem

Instituto Tecnológico de Aeronáutica (ITA); Universidade Federal de São Paulo (UNIFESP)
São José dos Campos, SP, Brazil
rmoliveira17@gmail.com

Guilherme Coelho

Lojas Renner S.A.
Porto Alegre, RS, Brazil
guilherme.coelho@lojasrenner.com.br

Luiz Leduino Salles Neto (Advisor)

Universidade Federal de São Paulo (UNIFESP)
São José dos Campos, SP, Brazil
luiz.leduino@unifesp.br

ABSTRACT

This study addresses the multiwave planning problem of reducing repeated stock-keeping units (SKUs) in omnichannel retail. It is motivated by high operational costs and inefficiencies caused by recurring SKU picks across different waves. Traditional approaches typically do not focus on limiting SKU overlaps across different waves, which can lead to excessive labor demands and delayed fulfillment. An exact model is proposed to partition boxes into homogeneous, capacity-constrained waves while minimizing the repetition of the SKUs. Empirical results from real instances at Lojas Renner S.A. reveal a notable drop in SKU overlap compared to existing operations, with all capacity requirements met. The findings indicate that systematic wave planning reduces redundant picking tasks and improves throughput, delivering measurable gains for large-scale fashion distribution centers.

KEYWORDS. Order batching; Multiwave planning; SKU overlap; Integer linear programming; Retail logistics.

Paper topics: Combinatorial Optimization; Logistics and Supply Chain Management.

1. Introduction

Omnichannel retail requires fast and accurate order fulfillment in physical and digital channels. In this setting, wave planning dictates how distribution centers group picking tasks into discrete production waves. Our partner, Lojas Renner S.A., incurs high labor and time costs when the same SKUs appear in multiple waves. SKU overlap forces pickers to revisit the same locations, reducing throughput.

Minimizing SKU overlap is critical because it directly impacts operational speed and cost. Traditional methods typically rely on manual grouping or simple heuristics that do not effectively handle overlapping SKUs among boxes. Due to their greedy nature, they can produce multiple waves with identical products, thereby increasing overall picking effort.

We address this by representing each box as a vertex in a graph, linking vertices with edges when they share SKUs, and then partitioning these vertices into a fixed number of sets (waves). Our integer-linear formulation enforces wave homogeneity and capacity limits while minimizing the number of SKUs appearing in multiple waves. We evaluate solutions by counting SKU overlap across waves while enforcing capacity.

This problem was studied in the context of Lojas Renner S.A., a fashion and lifestyle ecosystem with more than 600 stores in Brazil, Argentina, and Uruguay, in addition to digital channels. Founded in 1965, the company has become a leader in omnichannel fashion retail in Brazil, employing more than 21,000 employees.

Organization of the paper. The remainder of this paper is organized as follows. Section 2 reviews the related literature. Section 3 defines the operational setting, notation, and problem requirements. Section 4 details the preprocessing pipeline and the optimization of capacity-exceeding subgroups. Section 5 presents the exact integer programming model. Section 6 reports computational results on 135 instances and discusses performance indicators. Section 7 concludes, outlining limitations and directions for hybrid exact–heuristic extensions.

2. Literature Review

Many warehouse order picking studies focus on either time or cost objectives, but few studies rigorously examine the operational impact of SKU overlap across multiple waves. We examine four major works in the following, highlighting their methods, data scales, and validation practices. Then, we explain why none of the solutions resolves the challenge of repeated SKUs under multiwave constraints.

Rahmani Mokarrari et al. [Rahmani Mokarrari et al., 2025] develop two integer-programming frameworks—sequential versus simultaneous—for picker-to-parts systems. Their models capture practical complexities like heterogeneous pickers, item weights, and workforce limitations. They rely on real data from a North American consumer goods warehouse, underscoring the importance of empirical grounding. However, both frameworks focus on maximizing the total number of completed orders; they neither incorporate multiwave constraints nor account for the possibility that individual SKUs may appear in several waves.

Deng et al. [Deng et al., 2025] propose a 0–1 integer programming model for scattered storage batching and then enhance a seed batching approach with tabu search. Their study is notable for using large-scale, correlation-based grouping of orders to reduce travel distances. Despite discussing how correlated products can be beneficially clustered, their method essentially handles a single batch scenario at a time; it does not extend to penalizing repeated SKUs across multiple waves or enforce explicit wave homogeneity.

Xu et al. [Xu et al., 2025] apply an improved NSGA-II algorithm for multi-objective picking, balancing total picking time, order delay costs, picking costs, and workload distribution.

Their approach is validated through case studies, indicating real-world relevance. However, the core objective set—time minimization, cost reduction, and workload balance—does not include SKU overlap in separate waves. Although multi-objective, these objectives do not capture the specific penalty of repeatedly picking the same SKU over different waves.

Bock and Boysen [Bock and Boysen, 2025] present an exact routing approach for combined stowing and picking in parallel-aisle warehouses, exploiting polynomial-time complexity under certain structural constraints. They focus on merging two operations, stowing and picking, within a single tour, introducing nine distinct policies to reduce travel distance. While their model underscores the complexity of scattered storage, it omits multiwave planning entirely. Multiwave SKU overlap does not arise in their scenario, making it incompatible with the problem of partitioning SKUs across multiple waves.

Each work addresses batching or routing, yet none explicitly deals with repeated SKU picks across multiple waves. Rahmani Mokarrari et al. [Rahmani Mokarrari et al., 2025] and Deng et al. [Deng et al., 2025] both concentrate on single-phase allocations (maximizing total orders or clustering related orders, respectively), while Xu et al. [Xu et al., 2025] and Bock and Boysen [Bock and Boysen, 2025] focus on reducing picking distance, time, or cost within one continuous process. Because none of these methods penalizes SKU overlap between waves, opportunities to minimize extra handling or travel for recurring SKUs remain unexplored.

All four studies use real data sets, large-scale experiments, or both, attesting to the feasibility of their approaches. For instance, Rahmani Mokarrari et al. [Rahmani Mokarrari et al., 2025] analyze the effects of adding pickers or balancing their workload and conclude that more than 700 orders per shift demand additional workforce. Deng et al. [Deng et al., 2025] show that correlation-based seed batching reduces travel in high-volume environments. Xu et al. [Xu et al., 2025] validate multi-objective results with sensitivity analyses. Bock and Boysen [Bock and Boysen, 2025] benchmark multiple combined stowing-and-picking policies under typical e-commerce scales. Still, their reported metrics—number of orders handled, cost savings, or route distances—do not illuminate how each approach would perform if repeated SKUs were distributed across multiple waves.

Although these methods acknowledge real-world concerns like heterogeneous pickers or scattered storage, they do not show how to integrate repeated SKU reduction into everyday operations. In multiwave environments, distribution centers require an explicit model for splitting SKUs across parallel or sequential waves. None of the four sources describes a direct mechanism to handle wave-level SKU overlaps, nor do they indicate how to incorporate it into the picking schedule.

Thus, while past contributions are robust in modeling time, cost, or correlated product groupings, they do not systematically address SKU overlap under multiwave constraints. None of their methods directly penalizes picking the same product multiple times across different waves. This gap suggests that a specialized wave-partitioning approach—one that enforces SKU constraints and capacity homogeneity across waves—remains an unfilled need. Our work aims to remedy this shortcoming by targeting repeated SKUs directly, thus improving multiwave efficiency in modern omnichannel retail scenarios.

3. Problem Description

The organization of production waves refers to the process of grouping shipping boxes into batches (waves) according to operational rules. Each box corresponds to a customer order or part of it and contains a set of Stock Keeping Units (SKUs). We use *unit* to denote a single counted SKU (e.g., one garment). Wave capacity and box loads are measured in units. Two attributes are associated with each box: the **Wave Class**, which specifies the operational category of the order (e.g., store replenishment or e-commerce), and the **Inventory Type**, which distinguishes whether the products are packed as boxed units or as hanger units. A *wave* is defined as a collection of

boxes that share the same Wave Class and Inventory Type and are processed together in the picking system.

The problem addressed in this work consists of determining (i) the number of waves required and (ii) the assignment of each box to a wave. The decision must respect the following conditions:

- **Homogeneity of attributes:** All boxes within a wave must share the same Wave Class and the same Inventory Type.
- **Capacity limits:** Each wave is subject to a maximum number of units, with specific thresholds for boxed and hanger inventories.
- **Completeness:** All boxes in the planning horizon must be assigned to some wave.
- **Minimization of SKU overlap:** The allocation should minimize the occurrence of the same SKU in multiple waves, since each repetition requires additional retrieval from the storage location.

This definition captures the operational context of Lojas Renner S.A., where the objective is to reduce redundant picking of the same SKUs across waves, thereby improving the efficiency of order consolidation.

4. Preprocessing and Subgroup Optimization

The solution approach is divided into two stages. The first stage is a preprocessing procedure that applies deterministic rules to reduce the size of the problem and identify groups of boxes that can be directly assigned to waves. The second stage is an optimization step applied only to subgroups of boxes that exceed capacity limits and therefore require combinatorial allocation. This separation clarifies the role of preprocessing and highlights its impact on the subsequent exact model.

4.1. Preprocessing of Input Data

The input consists of a set of boxes, each associated with (i) a Wave Class, (ii) an Inventory Type, and (iii) the list of SKUs it contains. The preprocessing follows three steps:

1. **Initial grouping.** Boxes are clustered according to their Wave Class and Inventory Type, as illustrated in Figure 1a. The number of initial groups establishes a lower bound on the number of production waves, since boxes with different attributes cannot be mixed. Figure 1b shows an example with four such groups.
2. **Capacity check.** Each group is compared with the capacity thresholds (8000 units for *boxed inventory* and 6000 units for *hanger inventory*). If the total number of units does not exceed the threshold, the group forms a single wave, guaranteeing that all SKUs in that class/type are consolidated without repetition. Figure 2a illustrates this step.
3. **Subgrouping by shared SKUs.** If a group exceeds capacity, partitioned into SKU-based subgroups. Each subgroup contains boxes that share at least one SKU, which avoids unnecessary duplication when possible. Figure 2b provides an example of a feasible subgroup within capacity.

At the end of this stage, some groups are fully resolved (they can be directly assigned to waves), while others still exceed capacity and must be optimized in the next stage. Table 1 summarizes the outcomes for the example instance.

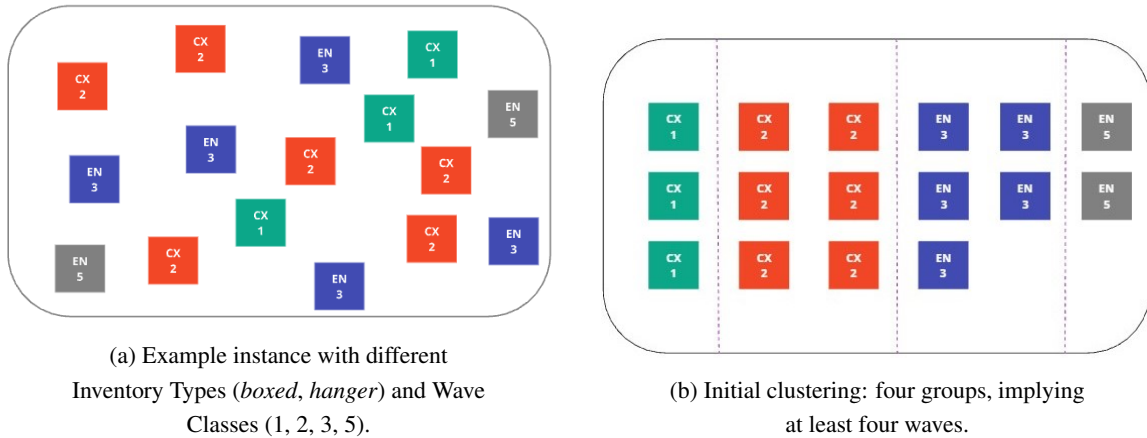


Figure 1: Illustration of the example instance and its initial clustering.

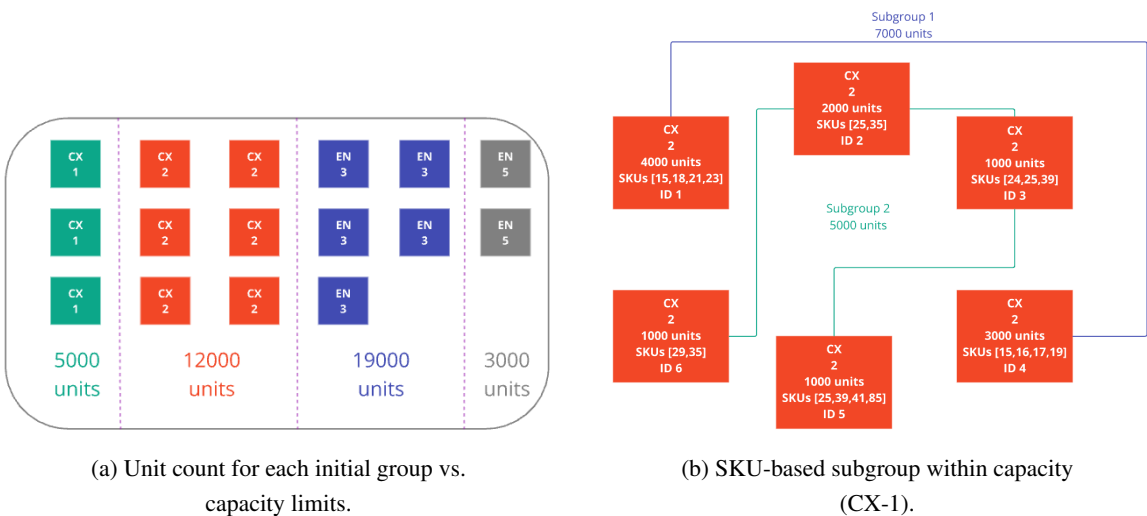


Figure 2: Capacity check and a feasible SKU-based subgroup.

4.2. Optimization of Capacity-Exceeding Subgroups

For subgroups whose total number of units still exceeds the capacity limits, a combinatorial decision problem arises. The objective is to partition the boxes into feasible waves while minimizing the number of repeated SKUs across waves. This step links the preprocessing stage with the exact optimization model.

Figure 3a illustrates the case of subgroup EN-3, which totals 9,000 units and exceeds the 6,000-unit capacity for hanger inventory. Several partitioning configurations are feasible, each with a different number of SKU overlaps. For example:

- Boxes 7 and 8 in one wave and box 9 in another: feasible, with 3 repeated SKUs.
- Boxes 7 and 9 in one wave and box 8 in another: feasible, with 4 repeated SKUs.
- Boxes 8 and 9 in one wave and box 7 in another: infeasible, as the total exceeds capacity.

The chosen configuration (boxes 7 and 8 together, box 9 separately) satisfies capacity while introducing the fewest repetitions. Boxes with no common SKUs, such as boxes 10 and

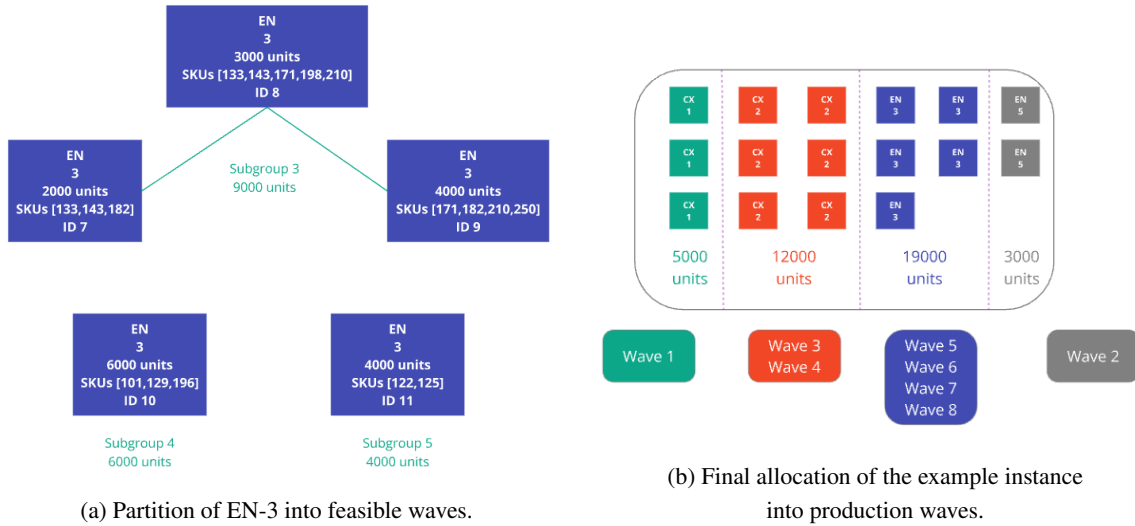


Figure 3: Capacity-exceeding subgroup partition (a) and final wave allocation (b).

11, are assigned individually to waves. The final allocation for the example instance is shown in Figure 3b.

Table 1: Summary of preprocessing decisions for the example instance.

Group / Subgroup	units	Capacity	Decision	SKU overlap
CX-1	5000	8000 (boxed)	One wave	0
EN-5	3000	6000 (hanger)	One wave	0
CX-2	7000	8000 (boxed)	One wave	0
Subgroup (Boxes 2,3,5,6)	5000	8000 (boxed)	One wave	0
EN-3 (Boxes 7,8,9)	9000	6000 (hanger)	Split into 2 waves	3
Box 10	2000	6000 (hanger)	Single wave	0
Box 11	1500	6000 (hanger)	Single wave	0

After preprocessing, two outcomes are obtained: (i) groups of boxes whose total number of units does not exceed capacity, which are directly assigned to waves; and (ii) residual subgroups of boxes that remain above capacity and must be optimized. The exact model presented in the next section is applied exclusively to these residual subgroups. Formally, let:

- I : set of boxes in a residual subgroup (output of preprocessing);
- S : set of SKUs contained in boxes $i \in I$;
- p_i : number of units in box $i \in I$;
- C : capacity of a wave (depends on the Inventory Type of the subgroup).

The preprocessing stage determines which boxes are excluded from I (those already fixed into waves) and which boxes compose I (those requiring optimization). This reduced instance is then passed to the exact integer programming formulation in Section 5, where the objective is to assign boxes to feasible waves while minimizing SKU overlap.

5. Exact Model

5.1. Sets and Parameters

- I : set of boxes in the cluster.
- S : set of SKUs (products) in the cluster.
- W : set of possible waves. The maximum number of waves $|W|$ is determined by the ceiling of the division of the total number of units by the capacity.
- p_i : number of units in box $i \in I$.
- s_i : SKU associated with box $i \in I$.
- C : maximum capacity (in units) of a wave.

5.2. Variables

Decision Variables:

- $x_{i,w} \in \{0, 1\}$: binary variable that indicates whether box $i \in I$ is assigned to wave $w \in W$.

Auxiliary Variables (derived from the decision variables):

- $y_{s,w} \in \{0, 1\}$: binary variable that indicates whether SKU $s \in S$ is present in wave $w \in W$.
- $n_s \in \mathbb{Z}_+$: number of waves in which SKU $s \in S$ appears.

5.3. Objective Function

Minimize the total number of SKU divisions among the waves:

$$\min \sum_{s \in S} (n_s - 1) = \left(\sum_{s \in S} n_s \right) - |S|$$

Since $|S|$ is constant, we can simplify the objective function to:

$$\min \sum_{s \in S} n_s$$

Substituting n_s in terms of $y_{s,w}$, the objective function becomes:

$$\min \sum_{s \in S} \sum_{w \in W} y_{s,w}$$

5.4. Constraints

1. Box Assignment to Waves

Each box must be assigned to exactly one wave:

$$\sum_{w \in W} x_{i,w} = 1, \quad \forall i \in I$$

2. Wave Capacity Constraint

The total number of units in boxes assigned to each wave must not exceed the capacity:

$$\sum_{i \in I} p_i \cdot x_{i,w} \leq C, \quad \forall w \in W$$

3. Definition of Auxiliary Variables

a) Presence of SKUs in waves:

A SKU is considered present in a wave if at least one box containing that SKU is assigned to that wave:

$$y_{s,w} = \max_{i \in I: s_i = s} x_{i,w}, \quad \forall s \in S, \forall w \in W$$

To linearize this expression, we impose the following:

$$y_{s,w} \geq x_{i,w}, \quad \forall s \in S, \forall i \in I \text{ with } s_i = s, \forall w \in W$$

b) Calculation of the Number of Waves per SKU:

$$n_s = \sum_{w \in W} y_{s,w}, \quad \forall s \in S$$

5.5. Variable Domains

$$\begin{aligned} x_{i,w} &\in \{0, 1\}, & \forall i \in I, \forall w \in W, \\ y_{s,w} &\in \{0, 1\}, & \forall s \in S, \forall w \in W, \\ n_s &\in \mathbb{Z}_+, & \forall s \in S. \end{aligned}$$

6. Computational Results

The computational experiments were conducted on instances provided by the interested company. All instances were solved with an exact model implemented in Python 3.10 using Gurobi 12.0.0. A total of 135 test instances were analyzed with a time limit of 2000 seconds.

Experiments were run on a computer with the following specifications:

- **Processor:** AMD Ryzen 9 5900X 12-Core Processor, 3.70 GHz
- **Installed RAM:** 128 GB
- **System Type:** 64-bit operating system, x64-based processor
- **Operating System:** Windows 10

The model solved all instances; among the 83 optimized cases, it achieved optimal or near-optimal solutions with low gaps.

Consolidated indicators. Table 2 consolidates dataset size and outcomes. Out of **135** instances, **52** were solved in preprocessing (no optimization required) and **83** required the exact model. Among the optimized cases, **48** were proven optimal and **35** ended suboptimal within the time limit. Final status by instance is **100 optimal** (=52+48) and **35 suboptimal**. We also report the number of SKU-based subgroups (“clusters”) associated with each final status.

Gap Analysis. Figure 4 shows a histogram of the gaps between the solution obtained by the exact model and the lower bound provided by the solver. In particular, 51 instances exhibit a gap close to 0 (or equal to 0).

Table 2: Dataset and outcome summary.

Metric	Count / Share	Notes
Instances (total)	135 (100.0%)	
Solved in preprocessing	52 (38.5%)	No optimization required
Required optimization	83 (61.5%)	Input to exact model
Optimized: proven optimal	48 (57.8% of 83; 35.6% of all)	Certificate of optimality
Optimized: suboptimal	35 (42.2% of 83; 25.9% of all)	Within time limit
Clusters in optimal instances	1,863	SKU-based subgroups
Clusters in suboptimal instances	57	SKU-based subgroups
Avg. clusters per optimal instance	18.6	1,863/100
Avg. clusters per suboptimal instance	1.6	57/35

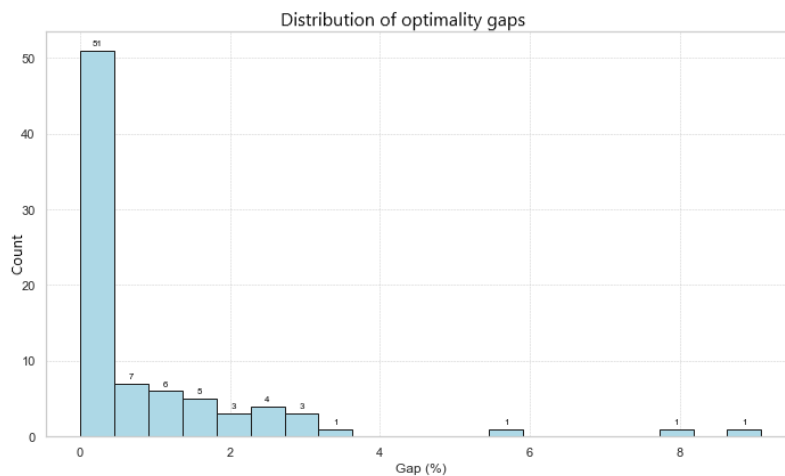


Figure 4: Histogram of gaps between the exact model solution and the lower bound.

Table 3: Performance summary for the 83 optimized instances.

Indicator	Value	Notes
Near-zero gap (including zero)	51 (61.4%)	from Figure 4
Positive gap	32 (38.6%)	within time limit
Solved within 100 s	$\geq 60\%$	from Figure 7
Time limit per instance	2,000 s	per experimental setup
Per-cluster cap	300 s	ensures bounded time per subgroup

Solver performance on optimized instances. Among the **83** optimized instances, **51** exhibit a near-zero optimality gap (including zero), i.e., **61.4%**. More than **60%** of all instances were solved within **100 s** (Figure 7). Table 3 summarizes these indicators.

Discussion. preprocessing eliminates the need for optimization in **38.5%** of the instances, which explains the overall runtime profile: only the remaining **61.5%** feed the exact model. Within the optimized set, the model proves optimality in **57.8%** and attains a near-zero gap in **61.4%**, indicating a strong formulation with tight bounds for most cases. The large discrepancy in cluster counts by final status (1,863 vs. 57) combined with the averages in Table 2 suggests a structural difference

between instance groups: optimal-status instances concentrate many small SKU-based subgroups that are either directly feasible or easily partitioned, whereas suboptimal instances involve fewer subgroups that remain harder under the capacity and SKU-overlap constraints. Finally, all optimized solutions improve the Commercial Index relative to the operational baseline, showing that the partitioning objective is aligned with the industrial KPI even when the solver returns a nonzero gap.

Commercial Index Comparison. Lojas Renner employs a commercial index to assess the efficiency of production waves based on the number of products repeated in different waves. The index is calculated by:

1. Determining the production waves.
2. Counting, for each product, the number of waves it appears.
3. We sum these counts for all products.
4. Dividing by the total number of unique products.

The optimal (or best) scenario is achieved when each product appears in only one wave, with an index of 1. Figure 5 compares the commercial indices of all 83 instances. The blue line represents the indices observed in Lojas Renner's current operational system, while the red line represents those obtained with the exact model. It is evident that the indices computed by the exact model are consistently closer to 1, indicating lower product repetition and, therefore, a more efficient allocation.

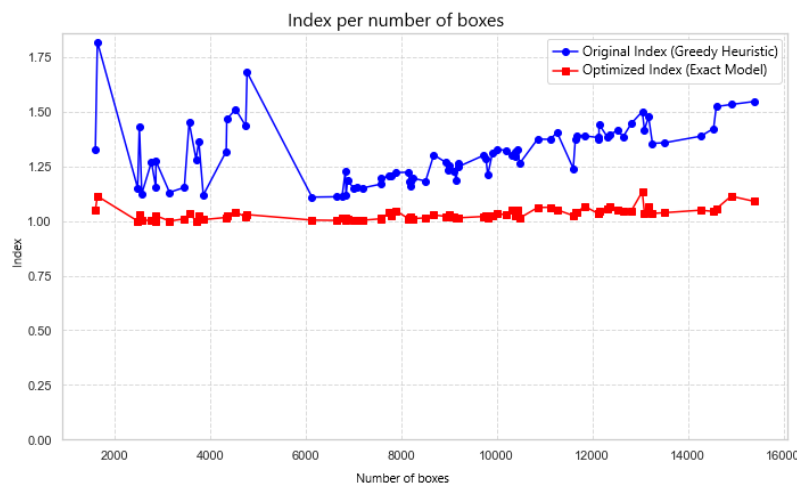


Figure 5: Comparison of the Commercial Index: Blue indicates the results achieved in practice by Lojas Renner, and red indicates the results from the exact model.

Instance Overview. Figure 6 summarizes the full dataset of **135 instances**. Of these, **52** were solved during preprocessing (no optimization required) and **83** required the exact model. Among the 83 optimized instances, the solver proved **48 optimal** solutions and returned **35 suboptimal** solutions within the time limit. Thus, the final status by instance is **100 optimal** (= 52 preprocessed + 48 optimized) and **35 suboptimal**. The bars labeled *Clusters* report the number of SKU-based

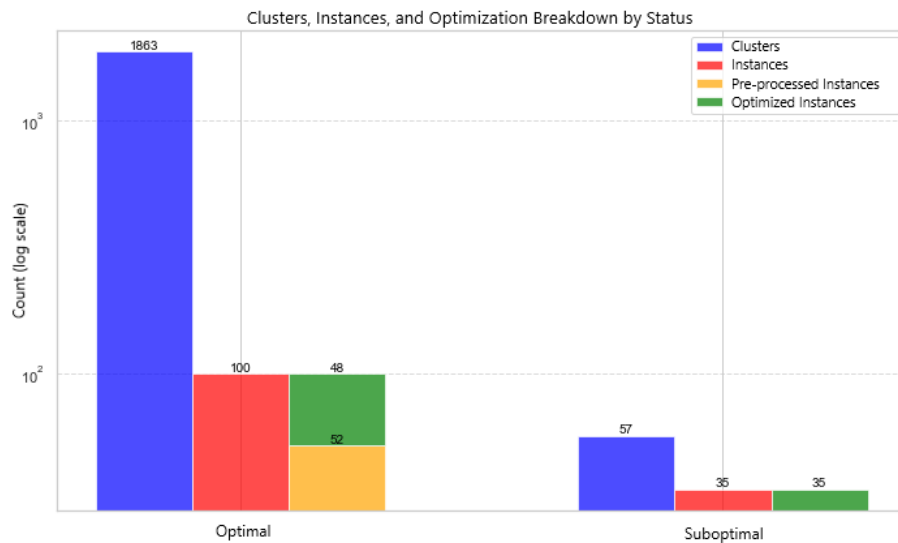


Figure 6: Overview of outcomes for 135 instances. Bars show counts by final status: *Instances* (100 optimal, 35 suboptimal), *Pre-processed Instances* (52), and *Optimized Instances* (48 optimal, 35 suboptimal). *Clusters* are SKU-based subgroups formed during preprocessing (1,863 in optimal instances; 57 in suboptimal ones). Numbers above bars are totals; y-axis in log scale.

subgroups created in preprocessing and associated with each status (**1,863** in optimal instances and **57** in suboptimal ones). Values above bars are absolute counts; the y-axis uses a logarithmic scale.

Figure 7 illustrates the execution times for each instance. Runtimes span up to the 2000 s cap; over 60 % of instances finish within 100 s. The execution time for each instance varies according to the volume and complexity of the clusters that require optimization; however, the processing time for each cluster was limited to 300 seconds, ensuring that even the more challenging clusters remained within this predefined per-cluster time cap.

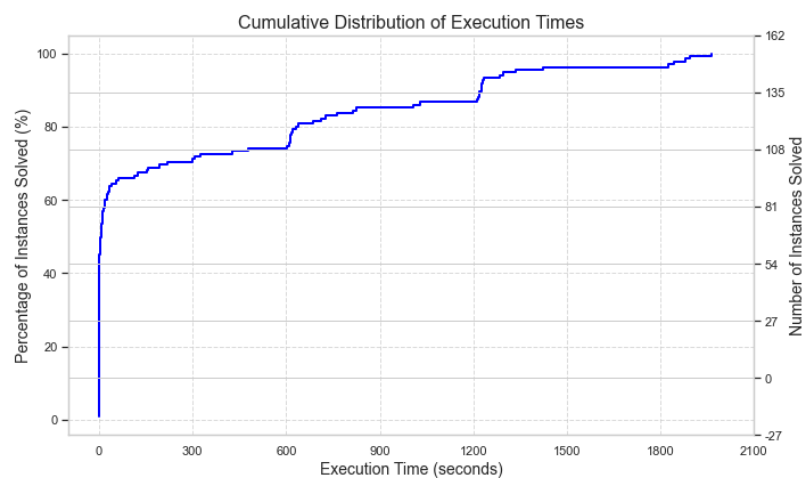


Figure 7: Execution time per instance.

Integrating the exact model into the production wave planning workflow yields significant improvements in both the Commercial Index and the number of production waves. A lower repetition of products across waves optimizes the picking process and potentially reduces overall

logistical and operational costs.

7. Conclusion

This study evaluated an exact optimization model for order batching in the production-wave organization at Lojas Renner S.A. The model achieved optimality on small instances and near-optimal solutions on larger ones with low optimality gaps. Compared with current operations, it consistently reduced the Renner Commercial Index—our proxy for SKU overlap across waves—indicating more efficient allocation of resources and lower operational costs.

Limitations. The formulation is single-objective, minimizing SKU overlap under deterministic wave capacities. It does not model travel distance, due dates, picker balancing, zoning/aisle constraints, or stochastic availability. Capacities are represented by a single scalar per inventory type and SKU weights are uniform. Planning is offline (no reoptimization after wave release). The pre-processing step enforces SKU-based subgrouping, which reduces problem size but may preclude cross-subgroup exchanges in rare cases. Time caps (2,000 s per instance; 300 s per subgroup) can yield suboptimal certificates for some instances.

Future work. Metaheuristic solvers can be devised for large instances. Promising directions include: (i) Iterated Local Search (ILS) with perturbation and acceptance rules guided by SKU-overlap deltas; (ii) Variable Neighborhood Search (VNS) alternating swap, relocate, and merge-split moves under capacity and homogeneity constraints; and (iii) GRASP, using a randomized feasible construction followed by intensive local improvement. Another avenue is Random-key Optimization, in which continuous keys encode box-wave assignments and a feasibility-preserving decoder enforces capacity and attribute constraints, enabling efficient neighborhood exploration. These methods can be benchmarked against the exact model on the same dataset using time-to-target and gap-to-best, and extended to multiobjective settings that trade off SKU overlap against the number of waves or proxies for travel time.

References

- Bock, S. and Boysen, N. (2025). Stow & pick: Optimizing combined stowing and picking tours in scattered storage warehouses. *European Journal of Operational Research*. Received 24 June 2024; Accepted 25 February 2025; Available online 7 March 2025. Open access under CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).
- Deng, Y., Jiang, W., Wang, Y., and Xu, B. (2025). Optimizing order batching and picking problems considering the correlation between products under the scattered storage mode. *Sustainability*, 17:1646. Academic Editor: Giada La Scalia. Received: 15 January 2025; Revised: 9 February 2025; Accepted: 10 February 2025; Published: 17 February 2025.
- Rahmani Mokarrari, K., Sowlati, T., English, J., and Starkey, M. (2025). Optimization of warehouse picking to maximize the picked orders considering practical aspects. *Applied Mathematical Modelling*, 137:115585. Received 24 July 2023; revised 18 June 2024; accepted 21 June 2024; available online 25 June 2024. CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).
- Xu, H., Shao, J., and Sun, Y. (2025). Research on order batching optimization based on improved nsga-ii algorithm. *PLOS ONE*, 20(2):e0319182. Published: February 25, 2025.