

A Flow Model to the Order Picking Problem

Iago Zagnoli Albergaria

Departamento de Ciência da Computação - Instituto de Ciências Exatas UFMG - Universidade
Federal de Minas Gerais
Av. Pres. Antônio Carlos, 6627 - Pampulha, Belo Horizonte - MG, 31270-901
iagozag@dcc.ufmg.br

Marcio Costa Santos

Departamento de Ciência da Computação - Instituto de Ciências Exatas UFMG - Universidade
Federal de Minas Gerais
Av. Pres. Antônio Carlos, 6627 - Pampulha, Belo Horizonte - MG, 31270-901
marciocs@dcc.ufmg.br

ABSTRACT

Order picking is one of the most critical operations in warehouse logistics, particularly in the context of e-Commerce, where speed and accuracy are paramount. This paper addresses the Order Picking in Waves problem, where a subset of customer orders is grouped for simultaneous collection to optimize operational efficiency. We propose a novel mathematical formulation based on network flow and introduce constraints that ensure the feasibility of selected order groups. Our objective is to maximize the average number of items picked per aisle, thereby reducing unnecessary traversal and improving productivity. We compare our flow-based formulation with a standard integer programming approach, highlighting its advantages in capturing item-aisle dependencies. Computational experiments in real-world-inspired instances demonstrate that our model produces high-quality solutions and supports a two-phase algorithm that efficiently finds the optimal number of aisles required. These results provide valuable insights for warehouse managers seeking to streamline order fulfillment operations in complex environments.

KEYWORDS. Flow. Optimization model. Order picking.

Paper topics. Order picking problem. Graph flow. Heuristics.

1. Introduction and motivation

Warehouses are customarily used to store products and are an important step in distribution chains for various products and distributors[Frazelle, 2002]. Thus, requiring intensive product handling operations[Baker and Canessa, 2009]. The most relevant of such operations in this context is the picking operation, and it is known to be one of the most machine intensive. Some studies, such as Tompkins et al. [2010], claim that the cost of such an operation corresponds to as much as 55% of the total warehouse operating expenses.

The picking problem in warehouses refers to the challenge of efficiently retrieving items from storage to fulfill customer orders. The warehouse pick-up problem is one of the most complex and cost-intensive operations in order fulfillment[Goetschalckx and Ashayeri, 1989], especially since online shopping increases the demand for speed, accuracy, and flexibility. It involves locating, retrieving, and moving items from storage to a packing or shipping area based on customer orders. In traditional retail, orders are picked in bulk for store delivery.

As online shopping continues to grow, warehouses must handle an increasing volume of small individualized orders with high speed and precision[Boysen et al., 2019]. This requires optimizing the picking process to minimize the time and distance that workers[Wruck et al., 2017] or automated systems spend collecting items[Lang and and, 2013]. Inefficiencies in picking can lead to delays, increased labor costs, and errors in order fulfillment, directly affecting customer satisfaction.

Order picking can be defined as the process of retrieving products from storage (warehouse) in response to specific customer requests. Once an order is placed by a customer, it enters a backlog of orders to be collected and is assigned a processing deadline based on the promised delivery time. The next step involves collecting the items from the warehouse that make up the order for subsequent packaging and shipping.

From an operational point of view, specially for large distributor, collecting items for individual orders is less efficient than doing so by grouping subsets of orders[Van Gils et al., 2018]. These groups of order subsets are called *order waves*, or simply *waves*. The main reason behind this fact is that collecting multiple orders at once allows one to take advantage of the synergies between the collecting locations of the items from different orders. The process of successively selecting the next subset of orders from the backlog to be collected, through one or more routes, is known as *order picking in waves*. Therefore, to maximize the productivity of the picking operation, the selection of the orders that will compose the next wave must be done carefully. Thus, the focus of this challenge is to propose algorithms to solve the problem of finding an optimal order wave that maximizes picking productivity.

In this paper, motivated by the problem proposed by Mercado Livre [2025], we studied the problem of order picking in waves. We propose a mathematical model based on flow that can be used to obtain good quality solutions. This paper is organized as follows. Section 2 presents the definitions and formal statement of the problem addressed in this work. Section 3 presents a standard formulation for the problem. Section 4 presents the proposed method and analyzes its applicability. Section 5 presents a two-phase algorithm designed to solve the problem and discusses its implementation. Section 6 presents the performance of the standard formulation and the two-phase algorithm on a comprehensive set of instances. The computational experiments show the compelling performance of both algorithms. And finally, Section 7 presents the conclusions and future research directions.

2. Problem Definition

We are given a set of orders in the backlog $\mathcal{O}$, where each order $o \in \mathcal{O}$ has a set of required items $\mathcal{I}_o$ needed to fulfill the order o . The set of all the necessary items for all orders is denoted by $\mathcal{I}$ and, for its definition, by $\mathcal{I} = \bigcup_{o \in \mathcal{O}} \mathcal{I}_o$. These items are stored in aisles within the warehouse, where $\mathcal{A}_i$ is the subset of aisles that contains at least one unit of the item i . The set of all aisles is denoted by $\mathcal{A}$ and $\mathcal{A} = \bigcup_{i \in \mathcal{I}} \mathcal{A}_i$.

In addition, we define the quantity of item $i \in \mathcal{I}$ requested by order $o \in \mathcal{O}$ as w_{oi} and, similarly, the quantity of item $i \in \mathcal{I}$ available in aisle $a \in \mathcal{A}$ is represented by z_{ai} . Finally, there are two other constants LB / UB to represent the lower and upper bound on the number of items picked, respectively.

The *Order Picking Problem*, a variant of the classical *Order Batching Problem*[Gademann and Velde, 2005], consists of determining a set of orders that must be grouped together in a wave so that the efficiency of the picking operation is maximized. In this case, the objective is to maximize the average number of items picked per aisle. More formally, let $\mathcal{O}' \subset \mathcal{O}$ be the subset of orders grouped together in a wave and $\mathcal{A}' \subset \mathcal{A}$ be the subset of aisles that must be visited to collect all the items required for the orders in $\mathcal{O}'$. It should be noted that this formulation presents only a framework for evaluating the feasibility and fitness of a solution given the subsets $\mathcal{O}'$ and $\mathcal{A}'$. The complete problem, including its decision variables, will be presented in the following section. In the wave picking system, the goal is to optimize the process of grouping orders and selecting aisles to maximize the efficiency of the picking operation. In this context, our objective is to maximize the average number of items collected per aisle, thus improving operational efficiency and reducing unnecessary movement within the warehouse. So, the function that maximizes the average of items per aisle can be mathematically expressed as follows:

$$\max \frac{\sum_{o \in \mathcal{O}'} \sum_{i \in \mathcal{I}_o} w_{oi}}{|\mathcal{A}'|} \quad (1)$$

The objective function (1) is what one expects. Furthermore, there are three main constraints to this problem that ensure that the solution remains feasible in terms of item collection and aisle capacity. They are stated below:

$$\sum_{o \in \mathcal{O}'} \sum_{i \in \mathcal{I}_o} w_{oi} \geq LB \quad (2)$$

$$\sum_{o \in \mathcal{O}'} \sum_{i \in \mathcal{I}_o} w_{oi} \leq UB \quad (3)$$

$$\sum_{o \in \mathcal{O}'} w_{oi} \leq \sum_{a \in \mathcal{A}'} z_{ai}, \forall i \in \mathcal{I}_o \text{ such that } o \in \mathcal{O}' \quad (4)$$

Constraints (2) and (3) establish limits on the total number of items collected in the wave, ensuring that the number of items falls within a specified range. The lower bound is important to maintain operational efficiency and avoid under-utilization of resources. In the other hand, the upper bound can be determined by storage capacity, processing limits, or any other logistical factor. Finally, the constraint (4) ensures that the aisles chosen contain enough items to fulfill all selected orders.

In this problem, an instance consists of the available orders and aisles. Each order is characterized by the number of items that it requests for each product, while each aisle is defined

Order	Item 0	Item 1	Item 2
0	2	0	0
1	1	4	1
2	0	0	3

Table 1: Orders and their items.

Aisle	Item 0	Item 1	Item 2
0	3	1	0
1	0	2	1
2	0	2	3

Table 2: Items in each aisle.

by the quantity of each item it contains. Tables (1) and (2) present the descriptions of the orders and aisles, respectively.

We still need the lower and upper bound to characterize a real instance of the problem. To show how these values can completely change the optimal solution, Table (3) describes four examples with different LB / UB values and their respective optimal choice of orders and aisles. For this instance, it can be demonstrated that the completion of all orders yields the optimal solution.

Example	LB / UB	Orders	Aisles	Total of items	Total of aisles	Objective value
0	1 / 2	{0}	{0}	2	1	2
1	2 / 3	{2}	{2}	3	1	3
2	4 / 4	$\emptyset$	$\emptyset$	0	0	Impossible
3	1 / 100	{0, 1, 2}	{0, 1, 2}	11	3	3.67

Table 3: Examples of lower and upper bounds in the current instance

In Example 3, the three items 0 of orders 0 and 1 are retrieved from aisle 0. The four items 1 of order 1 are retrieved from aisles 1 and 2, while the four items 2 from orders 1 and 2 are picked up at aisles 1 and 2. This results in 3, 4 and 4 items picked in aisles 0, 1, and 2, respectively, which leads to an average of 3.67 items picked per aisle. Other picking arrangements are possible for item 1 that would lead to the same objective value.

3. Standard Formulation

To effectively model the problem as an optimization model, it is essential to translate the orders and aisle sets into decision variables. These variables will indicate whether a specific order or aisle is selected as part of the optimal solution. The first variable, denoted as x_o , indicates whether a particular order $o \in \mathcal{O}$ is selected. Similarly, the second variable, denoted as y_a , is used to represent the inclusion of a specific aisle $a \in \mathcal{A}$ in the solution. The formal definition of both variables is presented as follows:

$$x_o = \begin{cases} 1, & \text{if } o \in \mathcal{O} \text{ is in the solution} \\ 0, & \text{otherwise.} \end{cases} \quad \forall o \in \mathcal{O}$$

$$y_a = \begin{cases} 1, & \text{if } a \in \mathcal{A} \text{ is in the solution} \\ 0, & \text{otherwise.} \end{cases} \quad \forall a \in \mathcal{A}$$

Once the variables are defined, we proceed to construct the objective function, which encapsulates the core goal of the problem. We can derive from the mathematical formulation in Section (2) the following objective function:

$$\max \frac{\sum_{o \in \mathcal{O}} \sum_{i \in \mathcal{I}} x_o w_{oi}}{\sum_{a \in \mathcal{A}} y_a} \quad (5)$$

However, not only is the above function nonlinear, but this function is also not convex, due to the division operation between the binary variable x_o in the numerator and y_a in the denominator. Therefore, to address this issue, we can apply a linearization technique to transform the objective function into a linear form, thereby making it possible to solve the model with standard linear optimization methods. Linearization is achieved by creating a new parameter k , which represents the number of aisles that will be used. After that, we can formulate the model as follows:

$$\max \quad \frac{\sum_{o \in \mathcal{O}} \sum_{i \in \mathcal{I}} x_o w_{oi}}{k} \quad (6)$$

$$st \quad \sum_{a \in \mathcal{A}} y_a = k \quad (7)$$

$$\sum_{o \in \mathcal{O}} \sum_{i \in \mathcal{I}} x_o w_{oi} \geq LB \quad (8)$$

$$\sum_{o \in \mathcal{O}} \sum_{i \in \mathcal{I}} x_o w_{oi} \leq UB \quad (9)$$

$$\sum_{o \in \mathcal{O}} x_o w_{oi} \leq \sum_{a \in \mathcal{A}} y_a z_{ai} \quad \forall i \in \mathcal{I} \quad (10)$$

$$x_o \in \{0, 1\} \quad \forall o \in \mathcal{O} \quad (11)$$

$$y_a \in \{0, 1\} \quad \forall a \in \mathcal{A} \quad (12)$$

The objective function (6) is just what one might expect. The constraint (7) explicitly sets the number of aisles to be used to the value of k , preserving the intended objective function in a linear form. The remaining constraints, from (8) to (10), have been discussed earlier in Section 2 and remain unchanged in this formulation. The constraints (11) and (12) are the domain of the variables.

To determine the optimal solution to our problem, we must consider the best possible mean number of items collected per aisle, given that we have selected k aisles from the available set. Let $mean_k$ represent the best solution when exactly k aisles are chosen. The optimal answer to our problem will be $\max(mean_k)$, $\forall k, k = 1 \dots n_{Aisles}$. This approach ensures that we identify the most efficient configuration by systematically exploring all feasible combinations of aisles.

4. Flow Formulation

In this section, we present a flow-based formulation to address the order picking problem. First of all, we present how to construct a network for a given instance of the order picking problem and then dive into a detailed explanation of its components and functionalities.

4.1. Flow network

Let G be a flow network graph with a source S and a sink T . The idea is to represent the relationships between orders, items, and aisles through a network of nodes and arcs, where each path from the source to the sink corresponds to a choice of an order.

First, the source node S is connected to each order node $o \in \mathcal{O}$ through an arc denoted s_o . The capacity of each arc s_o is given by the sum of the weights of all items requested in that order, mathematically represented as $\sum_{i \in \mathcal{I}} w_{oi}$. This ensures that the maximum flow leaving the source towards a specific order node does not exceed the total demand of that order. Once the flow reaches an order node, it must be directed to the corresponding items required by that order. Therefore, each order node has outgoing arcs to the item nodes, where the arc capacity between an order node and an item node i is given by w_{oi} . This configuration guarantees that the flow moving from an order to an item does not exceed the quantity needed.

The items themselves are linked to the aisles that store them. For each item node, there are arcs to the aisle nodes that contain that specific item, with each arc capacity defined by availability z_{ai} . This setting ensures that the number of items flowing from an item node to an aisle node respects the storage limit of the aisle. Finally, the aisle nodes are connected to the sink node T , and the capacity of these final arcs is calculated as the sum of the capacities of all items stored within each aisle, represented as $\sum_{i \in I} z_{ai}$. This construction ensures that the total flow through an aisle node does not exceed the cumulative capacity of items that can be stored in that aisle.

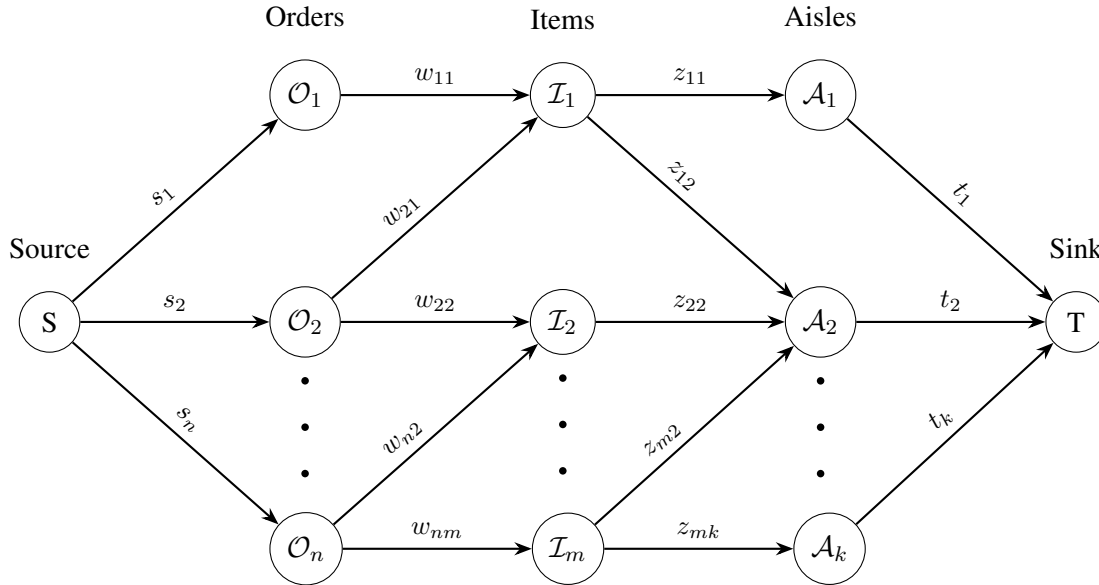


Figure 1: An example flow network of a fictional instance

Figure (1) illustrates a flow network for a fictional instance with n orders, m items, and k aisles. It is worth noting that if an order O_i does not request a given item I_j , for some $1 \leq i \leq n$ and $1 \leq j \leq m$, there is no need to include an arc from O_i to I_j with zero capacity. In other words, such an arc can be omitted. For example, in the figure, order 1 requests only item 1. Therefore, no arcs are drawn from order 1 to any other item. The same reasoning applies to arcs between items and aisles.

We still need to guarantee that if an order is selected to be part of the solution, all items associated with that order must be picked. This means that each edge s_o has only two possible states: either the flow through the edge is zero, indicating that the order is not part of the solution, or the flow is saturated, indicating that the entire order has been selected. To force this, we introduce a new constraint that establishes the relationship between the binary decision variable and the flow, formulated as follows:

$$s_o = x_o \sum_{i \in I} w_{oi} \quad (13)$$

Equation (13) ensures that if the binary variable x_o is equal to 1, indicating that the order is selected, then the flow s_o is equal to the total quantity of items required by that order. In contrast, if x_o is equal to 0, the flow through the edge s_o is also zero, effectively excluding the order from the solution.

4.2. Definitions

To provide further clarity, we define some essential elements of our model. The matrix $\mathbf{N}$ represents the incidence matrix of the graph $\mathbf{G}$, which captures the structure and connections between nodes and arcs. To simplify the model, we define $\mathbf{N}'$ as the incidence matrix excluding the rows corresponding to the source and sink nodes. The flow vector f contains variables that represent the flow amount for each arc of the graph, categorizing them based on their specific roles within the network. We denote the flow along the source arcs as f_s , representing the flow from the source to the order nodes. Similarly, f_w represents the flow between the order nodes and the item nodes, while f_z captures the flow between the item nodes and the aisle nodes. Finally, f_t denotes the flow from aisle nodes to the sink.

4.3. Model

$$\max \sum_{o \in \mathcal{O}} f_{s_o} \quad (14)$$

$$st \quad \mathbf{N}'f = 0 \quad (15)$$

$$f \leq \begin{pmatrix} s \\ w \\ z \\ t \end{pmatrix} \quad (16)$$

$$f_{s_o} = x_o \sum_{i \in I} w_{oi} \quad \forall o \in \mathcal{O} \quad (17)$$

$$\sum_{o \in \mathcal{O}} f_{s_o} \leq UB \quad (18)$$

$$\sum_{a \in \mathcal{A}} y_a = k \quad (19)$$

$$y_a \geq \frac{f_{t_a}}{t_a} \quad \forall a \in \mathcal{A} \quad (20)$$

$$f \geq 0 \quad (21)$$

$$x_o \in \{0, 1\} \quad \forall o \in \mathcal{O} \quad (22)$$

$$y_a \in \{0, 1\} \quad \forall a \in \mathcal{A} \quad (23)$$

The objective function (14) aims to maximize the total flow leaving the source, which corresponds to the total number of items picked within a wave. The constraint (15) ensures that the flow entering any vertex is equal to the flow leaving that vertex, which is essential to preserve the continuity of item collection across the aisles. Next, the constraint (16) forces an upper limit on the flow of any given edge, ensuring that the flow does not exceed the capacity assigned to that edge. The constraint (17) prevents partial order selection, ensuring that if an order is chosen, it must be selected in its entirety. Constraints (18) and (19) are identical to the corresponding constraints in the previous model discussed in Section (3). The constraint (20) enforces that if there is any positive flow leaving the aisle vertex, then y_a must be set to one, indicating that the aisle is indeed utilized. The constraint leverages the property that if f_{t_a} is greater than zero, the normalized value $\frac{f_{t_a}}{t_a}$ will also be positive. It is important to note that the fraction cannot exceed one, as constraint (16) guarantees that the flow on each edge does not exceed its capacity. Constraints from (21) to (23) are the domain of the variables.

Note that, unlike in the previous formulation, the model no longer includes a lower bound on the flow. However, since the objective function maximizes flow, this constraint is trivially satisfied as long as there is a feasible solution with K aisles and a maximum flow greater than LB .

5. Two phase algorithm

5.1. Number of Aisles x Optimal solution

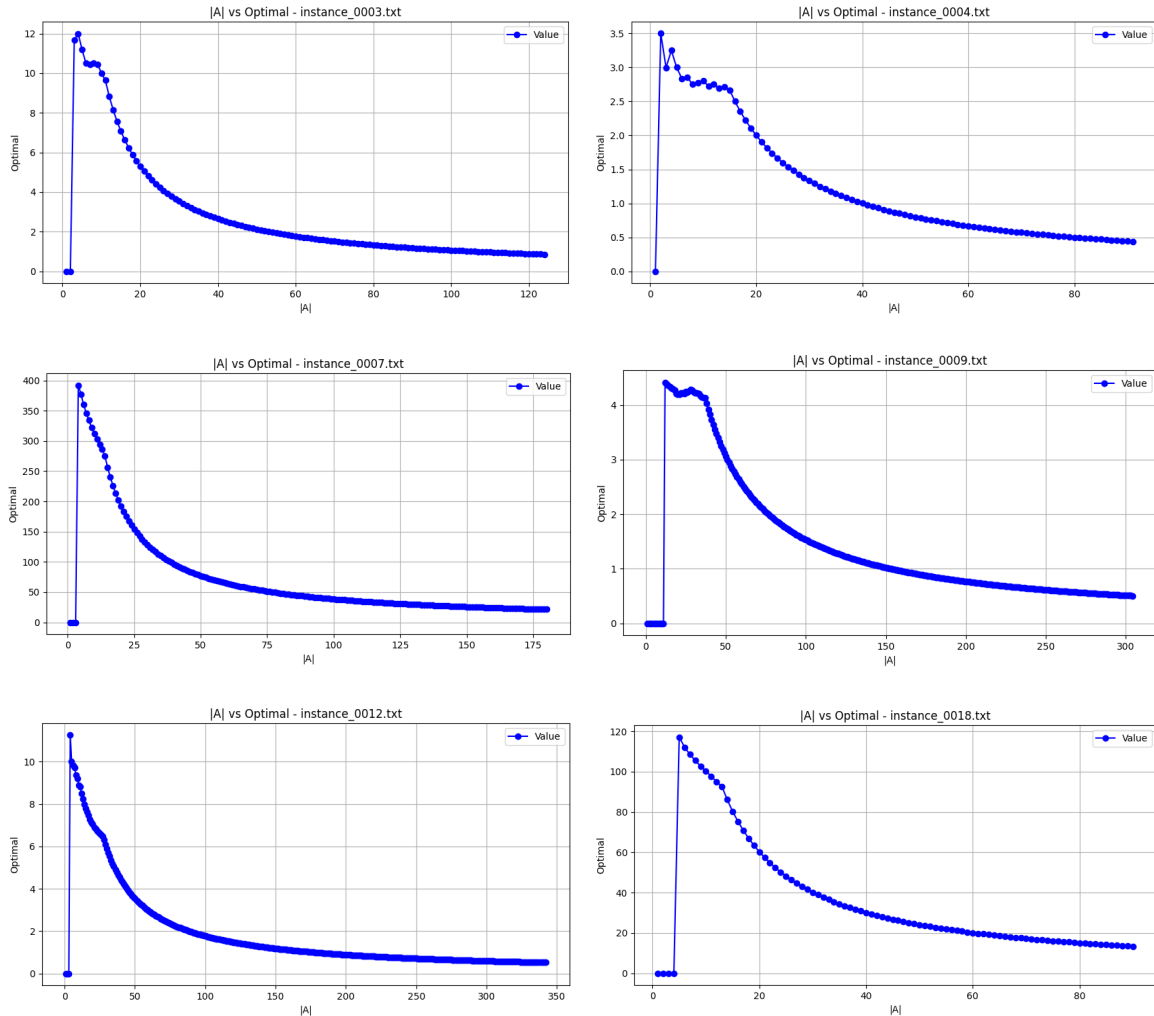


Figure 2: A comparison between the number of aisles and the optimal solution in six instances of dataset A, as available in Mercado Livre [2025]

Figure (2) presents six graphs, each illustrating the relationship between the number of aisles selected in a solution, denoted by $|A|$, and its corresponding optimal objective value. By comparing these two for a given instance, a pattern emerges. The left part of the graph corresponds to solutions that do not achieve a flow greater than the lower bound specified in the input. Once a feasible solution is found, where the number of items picked exceeds the lower bound (LB), the objective value either increases as additional aisles contribute to the solution or begins to decrease. When the objective value decreases, it rarely increases again, giving the impression that the function exhibits a unimodal behavior, even if it is not strictly unimodal.

In addition, it becomes evident that, in the majority of cases, the optimal solution is found in either the first viable solution or lies very close to it. This observation suggests that the search for the optimal number of aisles can be significantly reduced while still maintaining high accuracy. Therefore, an effective heuristic can be proposed by initially determining a lower bound of the number of aisles required.

5.2. First phase

Let L be the minimum number of aisles required to obtain a viable solution to the instance, i.e., there exists a solution using L aisles with a maximum flow greater than LB , and L is minimal. It is evident that running the model with fewer than L aisles is unnecessary, as no viable solution can be obtained in that case.

To efficiently determine the minimum number of aisles needed to achieve a viable solution, we can use a binary search. The objective is to find the smallest number L such that the maximum flow of the solution is greater than or equal to LB . By applying binary search over the interval $(1, nAisles)$, we can find this value in $O(\log nAisles)$ iterations, significantly reducing computational effort compared to a linear search. However, we can improve this process even further with a more direct approach. Instead of using binary search, we can determine the lower bound in just one iteration by modifying the flow model discussed in Section (4). We just need to change the objective function from maximizing the flow to minimizing the number of aisles while maintaining the feasibility of the solution. The new objective function can be formulated as follows:

$$\min \sum_{a \in \mathcal{A}} y_a \quad (24)$$

This new objective function (24) no longer aims to maximize the flow of selected items, but rather to minimize the number of aisles required to form a viable solution. However, simply minimizing the number of aisles is insufficient because we must also ensure that the chosen aisles provide enough flow to meet the required lower bound and that the variable y_a correctly represents whether aisle a is being used or not. To achieve this, we introduce two additional constraints that guarantee that the model satisfies the requirements of the problem.

$$\sum_{o \in \mathcal{O}} f_{s_o} \geq LB \quad (25)$$

$$y_a \leq f_{t_a} \forall a \in \mathcal{A} \quad (26)$$

The constraint (25) ensures that the total flow of the selected orders meets or exceeds LB , thus guaranteeing the viability of the solution. Constraint (26) is designed to link the binary variable y_a with the flow leaving the vertex corresponding to aisle $a \in \mathcal{A}$. This linkage ensures that y_a is zero if there is no flow leaving the aisle vertex, indicating that the aisle is not being used. If the aisle is not part of the solution, the binary variable appropriately reflects this by remaining zero. By combining the minimization objective with this constraint, this auxiliary model effectively finds the optimal L needed to complete the order picking process in a single iteration. This approach not only speeds up the computation, but also provides a precise and efficient way to determine the minimum number of aisles required.

5.3. Second phase

Once L is established, instead of exhaustively evaluating every possible configuration, we can limit the number of iterations by defining an iteration limit, denoted IT . Then, we can run the

model iteratively for all values from L to $L + IT$, systematically exploring configurations within this restricted range. By doing so, we focus only on the most promising configurations, rather than evaluating the entire search space, which would be computationally expensive.

6. Computational Results

In this section, we report on our computational experiences with the algorithms proposed in randomly generated instances. The algorithms were implemented in Java computer language and the experiments were performed on an AMD Ryzen 7 5700U and 8GB RAM personal computer. We used the package CPLEX 22.1.2 to solve linear programs. All experiments were conducted under the Arch Linux distribution with Hyprland as window manager.

To test our algorithm, we have adopted the set of test problems available in Mercado Livre [2025]. The test instances can be characterized by the following parameters:

- n = number of orders;
- m = number of items;
- k = number of aisles;
- LB = lower bound;
- UB = upper bound.

Instance	n	m	k	LB	UB	Obj value ST	Obj value TP	Optimal
a/01	61	155	116	30	68	15.00	15.00	15.00
a/02	7	7	33	0	2	2.00	2.00	2.00
a/03	82	246	124	33	106	12.00	12.00	12.00
a/04	16	59	91	4	40	3.50	3.50	3.50
a/05	2625	6407	161	1322	4395	177.88	177.88	177.88
a/06	10341	7089	184	452	3892	691.00	691.00	691.00
a/07	8320	5747	180	1306	3847	392.25	392.25	392.25
a/08	2185	5831	168	1304	2840	162.94	162.94	162.94
a/09	70	222	304	52	153	4.42	4.42	4.42
a/10	1602	3689	383	416	1746	X	13.45	17.11
a/11	1029	2784	375	330	2045	16.85	16.85	16.85
a/12	133	337	342	35	177	11.25	11.25	11.25
a/13	8375	7525	413	1510	4583	117.38	117.38	117.38
a/14	12402	10974	413	1947	7739	181.64	181.64	181.64
a/15	7367	6633	402	384	3679	149.33	149.33	149.33
a/16	1108	1051	88	149	686	85.00	85.00	85.00
a/17	417	411	83	54	175	36.50	36.50	36.50
a/18	2682	2309	90	537	1205	117.20	117.20	117.20
a/19	2257	2104	134	152	999	202.00	202.00	202.00
a/20	5	5	5	5	12	5.00	5.00	5.00

Table 4: Computational results for test instances of dataset A

Instance	n	m	k	LB	UB	Obj value ST	Obj value TP	Optimal
b/01	11321	7675	185	1206	3968	631.50	631.50	631.50
b/02	2162	5922	165	626	3405	155.67	180.64	180.64
b/03	2070	5617	164	585	2150	149.60	149.60	149.60
b/04	2448	6271	163	701	2802	150.40	150.40	155.67
b/05	6468	4959	179	996	2869	368.00	368.00	368.00
b/06	1857	4982	165	496	3558	72.71	77.14	83.97
b/07	1841	1621	165	91	833	108.00	108.00	108.00
b/08	12334	10365	398	2235	6120	203.36	227.10	227.10
b/09	5581	4990	417	1123	3205	81.07	82.50	82.50
b/10	14952	13510	482	1228	9361	434.33	444.00	444.00
b/11	45112	37820	482	8541	19101	X	776.45	898.80
b/12	28263	24432	483	6085	12808	X	507.67	567.64
b/13	38214	32604	483	6610	19417	X	600.91	756.78
b/14	38202	32238	483	4795	20406	X	599.38	885.17
b/15	11541	9905	398	2323	7207	X	145.75	166.14

Table 5: Computational results for test instances of dataset B

Tables (4) and (5) report the results of the 35 test instances provided in Mercado Livre [2025]. The first column identifies the test set, while the next five columns describe the characteristics of each instance. The column "Obj value ST" shows the best objective value obtained by the program using the standard formulation, whereas "Obj value TP" reports the best objective value achieved with the Two-Phase algorithm. If no feasible solution was found, the corresponding entry is marked with an 'X'. Finally, the column "Optimal" indicates the optimal average number of items per aisle for each test.

It is important to note that, in the challenge, the instances were meant to be solved within a 10-minute time limit. However, in the reported results, all tests were performed within a time limit of 20 minutes to allow for a more meaningful comparison between the algorithms.

7. Conclusion and Future Work

In this work, we investigate the problem of optimizing wave-based order picking in warehouses, a crucial yet resource-intensive task in modern logistics. Motivated by real-world challenges faced by companies such as Mercado Livre do Brasil, we proposed a novel flow-based optimization model that captures the structural complexity of the order-picking problem while allowing for efficient and accurate solutions.

We began by formalizing the problem, emphasizing the importance of maximizing the average number of items collected per aisle to improve operational efficiency. A standard integer programming formulation was presented and analyzed. To address the limitations associated with non-linearities in that model, we introduced a flow network formulation, which allows for more intuitive modeling of item movements and enables scalable optimization.

Our experiments demonstrate that the flow-based model not only preserves the correctness of the original problem but also allows for the development of effective heuristics. In particular, we observed that the optimal number of aisles used in a solution tends to be close to the minimal viable configuration, suggesting that a two-phase approach can significantly reduce computational overhead without sacrificing solution quality.

Future research could focus on extending the model to incorporate dynamic or stochastic elements, such as item replenishment, time windows, or variable travel times. In addition, a limitation of optimizing only the average number of items picked per aisle is that it tends to favor solutions where a large volume of items is concentrated in only a few aisles. In practice, this may lead to solutions where items distributed across multiple aisles are systematically excluded from waves. To address this issue, it would be valuable to consider additional performance metrics and develop strategies to mitigate such biases. Moreover, integrating routing constraints or coupling this approach with machine learning techniques to predict effective initial configurations could further enhance performance in large-scale real-time warehouse environments.

References

- Baker, P. and Canessa, M. (2009). Warehouse design: A structured approach. *European Journal of Operational Research*, 193(2):425–436. ISSN 0377-2217. URL <https://www.sciencedirect.com/science/article/pii/S0377221707011356>.
- Boysen, N., de Koster, R., and Weidinger, F. (2019). Warehousing in the e-commerce era: A survey. *European Journal of Operational Research*, 277(2):396–411. ISSN 0377-2217. URL <https://www.sciencedirect.com/science/article/pii/S0377221718307185>.
- Frazelle, E. (2002). *Supply Chain strategic*. McGraw-Hill/Irwin.
- Gademann, N. and Velde, S. (2005). Order batching to minimize total travel time in a parallel-aisle warehouse. *IIE Trans.*, 37(1):63–75.
- Goetschalckx, M. and Ashayeri, J. (1989). Classification and design of order picking. *Logistics World*, 2(2):99–106.
- Lang, G. and and, G. B. (2013). Economic performance and customer expectation in e-fulfillment systems: A multi-channel retailer perspective. *Supply Chain Forum: An International Journal*, 14(1):16–26. URL <https://doi.org/10.1080/16258312.2013.11517307>.
- Mercado Livre (2025). Challenge SBPO 2025. <https://github.com/mercadolibre/challenge-sbpo-2025>.
- Tompkins, J. A., White, J. A., Bozer, Y. A., and Tanchoco, J. M. A. (2010). *Facilities planning*. John Wiley & Sons.
- Van Gils, T., Ramaekers, K., Braekers, K., Depaire, B., and Caris, A. (2018). Increasing order picking efficiency by integrating storage, batching, zone picking, and routing policy decisions. *International Journal of Production Economics*, 197:243–261.
- Wruck, S., Vis, I. F., and and, J. B. (2017). Risk control for staff planning in e-commerce warehouses. *International Journal of Production Research*, 55(21):6453–6469. URL <https://doi.org/10.1080/00207543.2016.1207816>.