A new integer linear program and a grouping genetic algorithm with controlled gene transmission for joint order batching and picking routing problem

Felipe Furtado Lorenci, Santiago Valdés Ravelo
Institute of Informatics
Federal University of Rio Grande do Sul, Porto Alegre, Brazil
{fflorenci, santiago.ravelo}@inf.ufrgs.br

***Abstract*—Efficiently managing large deposits and warehouses is not an easy task. The amount of variables and processes involved from the moment a consumer purchases a single product until its receipt is quite considerable. There are two major problems involving warehouses processes: the order picking problem (OPP) and the order batching problem (OBP). The OPP aims to minimize the distance traveled by a picker while collecting a set of products (orders). The OBP seeks to assign orders to batches with a capacity limit in order to minimize the sum of distances traveled during the retrieving of products from all batches. When these two problems are approached together, they become the Joint Order Batching and Picking Routing Problem (JOBPRP). This work proposes a novel formulation for JOBPRP and develops a grouping genetic algorithm with controlled gene transmission. To assess our proposals, we executed computational experiments over literature datasets. The mathematical model was used within a mixed-integer programming solver (Gurobi) and tested on the smaller instances to evaluate the quality of the solutions of our metaheuristic approach. Our computational results evidence high stability for all tested instances and much lower objective value than the previously reported in the literature, while maintaining a reasonable computational time.**

***Index Terms*—Metaheuristic, Genetic Algorithms, Order Batching Problem, Order Picking Problem, Warehouses**

I. Introduction

The growth of internet services in the last decades boosted e-commerce (sales through websites and applications), that has been standing out as a sustainable, scalable and profitable business model. In 2021 the estimate movement of this sales model was around U$ 469.2 billion, only in the united states [7]. Statistics also point to a projection of U$ 563.388 billion for 2025 [3]. This huge market is based on attractive prices, ease and intuitiveness at purchase moment and short delivery times. In this way, to improve all logistics from sale to receipt, efficient warehouse management systems (WMS) interconnected with Enterprise Resource Planning (ERP) systems become indispensable for merchants that aims financial health on their companies.

Taking into account the current WMS, a major challenge within the warehouses is the dynamics that involves the picking of products purchased by customers, once studies indicates that about 55% of the operational costs in these places are related to this process [14]. As a result of this fact, practical combinatorial optimization problems arise, such as the Order Picking Problem (OPP) and the Order Batching Problem (OBP).

The OPP aims to minimize the distance traveled by a human or robot employee (pickers) in the aisles of a warehouse, during the picking of a product's list. OBP is an interconnected problem, that aims to group customer orders in batches to be assigned to the pickers, where each batch must satisfy a maximum capacity limit (weight/volume). The idea is to find a distribution of orders into batches, that minimizes the distance's sum of the routes traveled by each picker to retrieve the requested products. Both OPP and OBP are problems that belong to NP-Hard complexity class [5], [9].

Note that OBP depends on some routing metric to find possible order batches. When this problem is treated isolated, options to deal with distances may use order-picking routing policies (e.g., S-Shape, Largest Gap, Midpoint) or advanced heuristics. However, an alternative is to optimize individually the correspondent route of each batch during the batching process, that is, an approach that performs OBP jointly with OPP [17] - this strategy is known as Joint Order Batching and Picking Routing Problem (JOBPRP) [2].

A study of OBP was conducted by Henn and Wäscher [6], where two approaches based on the Tabu Search (TS) principle were proposed: a classic TS and an attribute-based hill climber (ABHC), achieving very competitive results. The instances' groups provided by these authors are used until nowadays as a standard benchmark in warehouse optimization problems. Koch and Wäscher [8] also explored OBP, developing a grouping genetic algorithm (GGA) that has been proven more performative than genetic algorithms (GA) approaches based on object-oriented encoding schemes. To the best of our knowledge, the results found by Menéndez, Pardo, Alonso-Ayuso, Molina and Duarte [12] with a Variable Neighborhood Search (VNS) methodology are the best to OBP, outperforming previous attempts in the state of the art.

Once that JOBPRP is a strategy to improve objective values of OBP through OPP, is possible to compare the results of both JOBPRP and OBP. In this sense, recently, Aerts, Cornelissens and Sörensen [1] approached JOBPRP on single-block warehouses, modeling the problem as a clustered vehicle

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

978-1-6654-6708-7/22/$31.00 ©2022 IEEE

routing problem (CluVRP) and solving it with a two-level VNS metaheuristic. They delivered good results compared with the literature, except with Menéndez et. al. [12] that maintains its superiority, mainly with larger batches capacities.

In this work we present a novel mathematical model and propose a grouping genetic approach to JOBPRP on single-blocks warehouses. Our model works independently on the warehouse layout (i.e., it could be multi-block, multi-floor or assume other formats beyond rectangular) and was used for solving smaller instances. To assess our metaheuristic proposal, we executed computational experiments over large datasets and our results evidence high stability and lower objective values when confronted with other studies.

The remainder of this paper is organized as follows: in the next section JOBPRP is formally defined. In section III our new mathematical formulation is presented. In section IV, we give a detailed explanation of our genetic approach. Section V shows the results of experimental tests, including comparisons with literature. In the last section, we discuss important points of this paper, as well as highlight difficulties faced during development and our future work directions.

II. Problem description

In this section we give an intuitive overview of JOBPRP, show examples of use, and formalize the problem definition. As discussed before, to reduce warehouses operational costs is a quite challenging task that involves many processes. We focus on those that occur at the dispatch area (also known as depot), where the manager receives orders from clients and assigns them to employees that will pick the specified consumer goods in the warehouse.

In this scenario, a warehouse layout is usually represented by a rectangular area divided into parallel aisles, which interconnect through two other cross aisles (one below and one above). The products are stored on shelves located to the right and left of each parallel aisle. Each shelve has an associated code, called **picking-position**, and any two shelves facing each other at the same aisle are considered to have the same picking-position. Figure 1 illustrates a warehouse with 5 parallel aisles, and some products on their shelves.

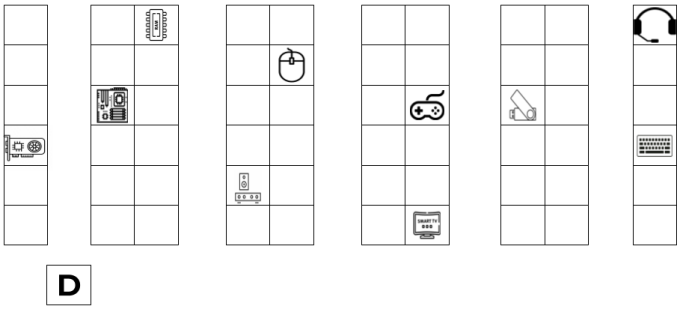

Fig. 1. An example of warehouse with highlighted products

Each product has an associated **address** defined by a pair ⟨aisle, picking-position⟩, that aims to identify the product's locations in the warehouse. In the example of Figure 1, if we consider the parallel aisles numbered from left to right, beginning at 0 and ending at 4, and the picking-position at each aisle from bottom to top, beginning at 0 and ending at 6, then the graphic card is located at address ⟨0,2⟩, while the video-game is at address ⟨3,3⟩, the same address of the Pen Drive. In the same figure, the location **D** denotes the depot, placed at the interception between the aisle 0 and the inferior cross-aisle; thus, its address is ⟨0,-1⟩.

An **order** is a collection of pairs ⟨product, address⟩ (examples on Table I), and represents goods to be retrieved. A **batch** is a set of orders. The orders are indivisible, i.e., all the products specified in an order must be allocated to a same batch. Each batch is assigned to a picker to collect the requested products.

TABLE I
ORDERS EXAMPLE

Orders			
Order 1		**Order 2**	
Product	**Address**	**Product**	**Address**
Headset	⟨4,5⟩	RAM Memory	⟨1,5⟩
Keyboard	⟨4,2⟩	Pen Drive	⟨3,3⟩
Bluetooth Mouse	⟨2,4⟩		
Pen Drive	⟨3,3⟩		
Order 3		**Order 4**	
Product	**Address**	**Product**	**Address**
Soundbar	⟨1,1⟩	Motherboard	⟨0,3⟩
Smart TV	⟨3,0⟩	Graphic card	⟨0,2⟩
Headset	⟨4,5⟩	Video-game	⟨3,3⟩
		Pen Drive	⟨3,3⟩
Order 5		**Order 6**	
Product	**Address**	**Product**	**Address**
Headset	⟨4,5⟩	RAM Memory	⟨1,5⟩
Pen Drive	⟨3,3⟩	RAM Memory	⟨1,5⟩
Keyboard	⟨4,2⟩		

The path traveled by the picker during the batches' products retrieving is known as **picking-route**, defined by a list of pairs ⟨label, address⟩, where **label** can be a product to be picked or the depot. Each picking-route has an associated distance equals to the sum of length units between each pair of consecutive addresses in its list. Every picking-route starts and ends at the depot. Each product has an associated weight, and since pickers have a capacity limit (weight/volume), each batch also needs a load bound. We define **c-capacitated picking-route** as a picking-route with the maximum capacity equal to c (i.e., the weight's sum of all products picked at the route should be less than or equal to c).

To guarantee the existence of feasible solutions, we assume that the total weight of an order does not exceed the capacity limit of a picker. Besides, we suppose that when a product is contained in an order, its availability is guaranteed on the stock. Note that some products can be requested in multiple orders, as well as some orders may demand more than just a single unit of a product.

To exemplify the JOBPRP, consider the orders in Table I and assume that each item has a weight 1 and the capacity limit of each picker is equal to 6. The task is to group the orders so that the total weight of each batch does not exceed 6, as this

is the maximum load that a picker can carry on. Hence, each batch has a corresponding 6-capacitated picking-route, and the objective is to minimize the sum of its distances.

A possible solution to the problem is the following batches configuration:

- **Batch 1:** order 1 and order 2
- **Batch 2:** order 3 and order 6
- **Batch 3:** order 4
- **Batch 4:** order 5

Furthermore, feasible options to 6-capacitated picking-routes are:

- **Batch 1:** Depot, ⟨0,-1⟩ → Ram Memory, ⟨1,5⟩ → Bluetooth Mouse, ⟨2,4⟩ → Pen Drive, ⟨3,3⟩ → Headset, ⟨4,5⟩ → Keyboard, ⟨4,2⟩ → Depot, ⟨0,-1⟩
- **Batch 2:** Depot, ⟨0,-1⟩ → Soudbar, ⟨1,1⟩ → RAM Memory, ⟨1,5⟩ → Headset, ⟨4,5⟩ → Smart TV, ⟨3,0⟩ → Depot, ⟨0,-1⟩
- **Batch 3:** Depot, ⟨0,-1⟩ → Graphic Card, ⟨0,2⟩ → Motherboard, ⟨0,3⟩ → Video-game and Pen Drive, ⟨3,3⟩ → Depot, ⟨0,-1⟩
- **Batch 4:** Depot, ⟨0,-1⟩ → Pen Drive, ⟨3,3⟩ → Headset, ⟨4,5⟩ → Keyboard, ⟨4,2⟩) → Depot, ⟨0,-1⟩

However, thinking about distance minimization, is the above solution an optimal batches-routes assignment? This is the central question that JOBPRP aims to answer, which is formally defined below.

Problem 1: Joint Order Batching and Picking Routing Problem JOBPRP

Input: a tuple $\langle n, c, \mathcal{E}, \omega, \phi, \delta \rangle$, where:

- $n \in \mathbb{N}$ indicates the total number of orders;
- $c \in \mathbb{N}$ indicates the maximum weight capacity of the batches;
- $\mathcal{E}$ is the warehouse addresses set. The depot is contained on $\mathcal{E}$. We denote as $|\mathcal{E}|$ the total number of warehouse addresses;
- ω is a vector, where ω_o represents the weight's sum of the products in the o-th order ($1 \leq o \leq n$);
- ϕ is a vector, where $\phi_o : \mathcal{E} \rightarrow \{0, 1\}$ is a binary function that assumes value 1 if address $i \in \mathcal{E}$ is required at the o-th order, and 0 otherwise ($1 \leq o \leq n$);
- $\delta : \mathcal{E} \times \mathcal{E} \rightarrow \mathbb{R}$ is a function that returns the distance between a pair of addresses $i, j \in \mathcal{E}$.

Output: a set of c-capacitated picking-routes, such that each order is completely contained in exactly one picking-route, and the sum of all picking-routes' distances is minimized.

In the next section, we introduce a novel mathematical model to JOBPRP.

III. Mathematical formulation

We propose a new integer linear program (ILP) for JOBPRP. We use this model within an exact solver to solve small instances, which also helps to estimate the accuracy of the algorithm proposed by us in the next section. From now on, we consider an instance $I = \langle n, c, \mathcal{E}, \omega_o, \phi_o, \delta \rangle$ of JOBPRP.

Our formulation is defined over two sets of binary variables and one of integer variables. The first set of binary variables controls the assignment of orders to batches. Once there are n orders (labeled from 1 to n), and each order is inserted in exactly one batch, the minimum number of batches necessary to satisfy the problem is 1 (when all orders are assigned to the same batch) and the maximum number is n (when each order is assigned to a different batch). To simplify notation, we define the set $\mathcal{N} = \{m \in \mathbb{N} | 1 \leq m \leq n\}$, and the variables x_{ob} are declared for each $o \in \mathcal{N}$ and $b \in \{m \in \mathbb{N} | 1 \leq m \leq o\}$. We enforce $b \leq o$ to avoid some symmetric solutions in which the batches are interchangeable:

$$x_{ob} = \begin{cases} 1, & \text{if order } o \text{ is in batch } b \\ 0, & \text{otherwise.} \end{cases}$$

The second set of binary variables describes the picking-routes, declaring a variable for all $i, j \in \mathcal{E}$ ($i \neq j$), $b \in \mathcal{N}$:

$$z_{ijb} = \begin{cases} 1, & \text{if } i \text{ is visited immediately before } j \text{ in the} \\ & \text{picking-route of batch } b \\ 0, & \text{otherwise.} \end{cases}$$

The integer variables indicate the ordinality of the addresses visited in the batches' picking-routes. If the address $i \in \mathcal{E}$ is visited by the picking-route of batch $b \in \mathcal{N}$, then u_{ib} indicates the ordinal position (starting at the depot) in which b visits i. These variables take value from 1 to $|\mathcal{E}|$ (there are no more than $|\mathcal{E}|$ addresses), and they are mainly used to avoid subtours in a feasible solution.

The problem's objective is to minimize the sum of all picking-routes' distances, which can be written as follows:

$$\min \sum_{b=1}^{n} \sum_{i \in \mathcal{E}} \sum_{\substack{j \in \mathcal{E} \\ j \neq i}} \delta(i, j) z_{ijb}$$

To guarantee that each order is assigned to exactly one batch, we define the set of constraints below:

$$\sum_{b=1}^{o} x_{ob} = 1 \quad \forall o \in \mathcal{N} \tag{1}$$

To ensure that the capacity limit of each batch is not exceeded, we state the following group of constraints:

$$\sum_{o=b}^{n} \omega_o x_{ob} \leq c \quad \forall\ b \in \mathcal{N} \tag{2}$$

The next two constraints groups guarantee for all addresses and batches, that if an address i is not requested in some of the orders of batch b, then i cannot precede nor succeed any other address in the picking-route of b:

$$\sum_{\substack{j \in \mathcal{E} \\ j \neq i}} z_{jib} \leq \sum_{o=b}^{n} \phi_o(i) x_{ob} \quad \forall\ b \in \mathcal{N},\ i \in \mathcal{E} \tag{3}$$

$$\sum_{\substack{j \in \mathcal{E} \\ j \neq i}} z_{ijb} \leq \sum_{o=b}^{n} \phi_o(i) x_{ob} \quad \forall\ b \in \mathcal{N},\ i \in \mathcal{E} \tag{4}$$

Also, we need to guarantee that every address has at most one predecessor and one successor at each picking-route:

$$\sum_{\substack{j\in\mathcal{E}\\ j\neq i}} z_{jib} \leq 1 \quad \forall\ b \in \mathcal{N},\ i \in \mathcal{E} \tag{5}$$

$$\sum_{\substack{j\in\mathcal{E}\\ j\neq i}} z_{ijb} \leq 1 \quad \forall\ b \in \mathcal{N},\ i \in \mathcal{E} \tag{6}$$

To include an address in a picking-route, we need to assure for all addresses, orders and batches, that if an address i is requested in at least one of the orders in batch b, then i must be preceded and succeeded by another address in the picking-route of b:

$$\sum_{\substack{j\in\mathcal{E}\\ j\neq i}} z_{jib} \geq \phi_o(i)x_{ob}\ \forall\ b \in \mathcal{N},\ b \leq o \leq n,\ i \in \mathcal{E} \tag{7}$$

$$\sum_{\substack{j\in\mathcal{E}\\ j\neq i}} z_{ijb} \geq \phi_o(i)x_{ob}\ \forall\ b \in \mathcal{N},\ b \leq o \leq n,\ i \in \mathcal{E} \tag{8}$$

Another important issue is to avoid sub-routes on picking-routes. Thus, we included the next constraints' set, based on Miller-Tucker-Zemlin (MTZ) [13] formulation of the traveling salesman problem (TSP):

$$u_{jb} \geq u_{ib} + 1 - |\mathcal{E}|(1 - z_{ijb})\ \forall b \in \mathcal{N}, i \neq j \in \mathcal{E} \setminus \{0\} \tag{9}$$

The following constraints aim to reduce symmetric solutions and establish a direction to the picker. Observe that, for each batch a same picking-route can be traveled in two different directions, hence we can reduce the search space by enforcing feasible solutions to only use one of those directions. For that, we define a sorting function $\sigma : \mathcal{E} \to \mathbb{N}$ which associates a different natural index to each address, being 0 the depot's index (i.e., $\sigma(d) = 0$). Since all picking-routes start at the depot, our proposal is to follow the direction where the second address will have lower σ value:

$$\sum_{i\in\mathcal{E}\setminus\{d\}} z_{dib}\sigma(i) \leq \sum_{i\in\mathcal{E}\setminus\{d\}} z_{idb}\sigma(i) \quad \forall\ b \in \mathcal{N} \tag{10}$$

Finally, we focus on breaking batch symmetries. Note that exchanging the total content from a batches' pair does not affect the objective value, and if there are empty batches, then equivalent solutions are obtained by swapping any used batch with an empty one. Thus, the following constraints' set enforces the usage of the first batches, guaranteeing that empty batches will be the last ones of a feasible solution:

$$\sum_{o=b}^{n} x_{ob}n \geq \sum_{o=b+1}^{n} x_{o(b+1)} \quad \forall\ 1 \leq b \leq n-1 \tag{11}$$

The above discussion lead us to the following model for JOBPRP:

Model 1: Integer linear program for instance $\langle n, c, \mathcal{E}, \omega_o, \phi_o, \delta\rangle$ of JOBPRP:

$$\min \sum_{b=1}^{n}\sum_{i\in\mathcal{E}}\sum_{\substack{j\in\mathcal{E}\\ j\neq i}} \delta(i,j)z_{ijb}$$

$$s.t.\ constraints\ (1)-(11)$$

$$x_{ob},\ z_{ijb} \in \{0,1\}, u_{ib} \in \mathbb{N}$$

IV. Grouped Genetic Algorithm

Our metaheuristic approach to JOBPRP is called "Grouping Genetic Algorithm with Controlled Gene Transmission for Joint Order Batching and Picking Routing Problem" (GGACGT-JOBPRP). We choose the grouping genetic algorithm (GGA) schema instead of an object-oriented GA (OO-GA) because Falkenauer demonstrated several advantages from GGA over OO-GA on Bin Packing Problem (BPP) [4]. Notice that the BPP seeks to minimize the number of bins (batches) used to package a set of items (orders) with different weights, such that the weight's limit of bins (maximum pickers capacity) is not exceeded. Consequently, BPP is the particular case of JOBPRP when all distances are the same.

A remarkable feature of GGA is the chromosome, which by definition is a set of groups that contains minor elements without repetition (the solution is a partition set). This representation implies in exclusion of symmetric solutions, allowing a faster convergence to the expected objective value [4]. Focusing on JOBPRP, the solution's groups (i.e., the chromosomes' genes) can be interpreted as the batches, and their contents are the orders. Also, each solution's batch has an associated picking-route.

Thus, considering that a chromosome encodes the orders distribution into batches, we summarize the main algorithm's pseudo-code at Algorithm 1, which receives an instance $\langle n, c, \mathcal{E}, \omega, \phi, \delta\rangle$ of JOBPRP and returns the best chromosome found β at line 30.

An empty population $\mathcal{P}$ of individuals is initialized at line 2. This population is filled with POP_SIZE members through an heuristic procedure (line 4) that is further detailed by Algorithm 2. The fitness' evaluation of the new individuals is performed during the generation process. The fitness value is equal to the objective value, which is equivalent to the sum of picking-route's distances. These distances are optimal, and are calculated through a dynamic programming approach proposed in [16]. Every time we evaluate the fitness, the distances traveled in each batch are stored, since these information are required at crossover. After the initialization process, the most adapted individual from $\mathcal{P}$ is assigned to β (line 7), the best solution found, which is updated whenever a better solution is found between lines 14-17 and lines 23-25.

Between lines 8 and 29 the algorithm executes ITERATION_LIMIT times the procedures that are described next. First, PARENTS_SIZE individuals from $\mathcal{P}$ are selected to be the parents $\mathcal{G}$ (line 9) of the next generation. These parents are chosen with a discrete probability distribution proportional to the ranking provided by the fitness. The resulting offspring

```
   Input  : an instance ⟨n, c, E, ω, φ, δ⟩ of JOBPRP
   Output : β, best chromosome found
1  begin
2      P ← ∅;
3      for i ← 1 to POP_SIZE do
4          C ← generateIndividual(n, c, E, ω, φ, δ);
5          P ← P ∪ {C};
6      end
7      β ← most adapted individual from P, considering F(β);
8      for t ← 1 to ITERATION_LIMIT do
9          G ← selectParents(P, PARENTS_SIZE);
10         Z ← crossover(G, n, ω, c);
11         for each x ∈ Z do
12             F(x) ← x's fitness;
13         end
14         β_z ← best solution of Z ;
15         if F(β_z) < F(β) then
16             β ← β_z;
17         end
18         for x ∈ Z \ {β_z} do
19             m ← mutation(x, ω);
20             if m is valid then
21                 update x's chromosome;
22                 F(x) ← x's fitness;
23                 if F(x) < F(β) then
24                     β ← x;
25                 end
26             end
27         end
28         P ← best POP_SIZE solutions from P ∪ Z;
29     end
30     return β;
31 end
```

Algorithm 1: Main GGACGT-JOBPRP approach

from the parents' crossover is assigned to the set $\mathcal{Z}$ (line 10), which is evaluated between lines 11 and 13. To ensure variability without losing genetic characteristics over generations, all individuals, excepting the best individual from $\mathcal{Z}$, can be mutated. The mutation process involves exchanging a pair of orders from different random batches, which may cause a batch overload (phenomena defined as **invalid mutation**). Aiming feasible solutions, if an invalid mutation is detected, it is discarded (line 20). Finally, at line 28 we define the new population as the best POP_SIZE solutions from the offspring $\mathcal{Z}$ and the current population $\mathcal{P}$.

In the subsections IV-A and IV-B, we detail, respectively the **generateIndividual** (generation of new individuals) and **crossover** procedures.

A. Individual generation

The individual generation heuristic described by Algorithm 2, aims to provide an initial population with lower objective function values. Therefore, its main idea is to group the most similar orders into the same batches, decreasing the associated picking-route.

The algorithm receives as input an instance $\langle n, c, \mathcal{E}, \omega, \phi, \delta \rangle$ of JOBPRP, and returns a new feasible solution $\mathcal{I}$. It starts creating an empty individual $\mathcal{I}$, and a set $\mathcal{O}$, that contains the orders (numbered from 1 to n) which are not allocated in solution (lines 2 and 3). At line 4, we assign the value 0 to the variable i, which represents the current batch that we are accessing.

```
   Input  : an instance ⟨n, c, E, ω, φ, δ⟩ of JOBPRP
   Output : a new individual I
1  begin
2      I ← ∅;
3      O ← {1, 2, ..., n};
4      i ← 0;
5      while |O| > 0 do
6          b ← new batch with index i;
7          r ← remove random order from O;
8          b ← b ∪ {r};
9          w ← total weight of b;
10         while some order of O fits in b do
11             s ← o ∈ O s.t min. OSD(b, o, δ, φ, E), w + ω_o ≤ c;
12             if s ≠ ∅ then
13                 O ← O \ {s};
14                 b ← b ∪ {s};
15                 w ← total weight of b;
16             end
17         end
18         I ← I ∪ {b} ;
19         i ← i + 1;
20     end
21     return I;
22 end
```

Algorithm 2: Individual generation algorithm

Between lines 5 and 20, we execute the main loop of the algorithm until all orders are included in some batch. At each iteration, we initialize a new batch b with index i (line 6) and choose a random order r to be removed from $\mathcal{O}$ and be pushed to b (lines 7 and 8). The total weight of b's orders is stored in the variable w (line 9). While some order from $\mathcal{O}$ fits in b (line 10), we assign to s the order o from $\mathcal{O}$ which minimizes OSD (Orders similarly degree) function without exceed the capacity limit c (line 11). If no order satisfies the load constraint, then s becomes empty. If s is not empty (line 12), then we remove it from $\mathcal{O}$, push it into b (lines 13 and 14) and update w at line 15. When the batch construction is concluded, the batch b is appended to solution (line 18).

The insight of using an OSD function is to determine which order $o \in \mathcal{O}$ generates the minimum distance for the picking-route when combined with the orders already in b. Hence, the OSD of an order $o \in \mathcal{O}$ consists of applying the dynamic programming algorithm of [16] to compute an optimal picking-route for retrieving all products of the orders in $b \cup \{o\}$.

B. Crossover procedure

The crossover algorithm applies a controlled gene transmission technique, that provides gains over other crossover methods [15]. At the end of this process, the batches with more orders and shorter picking distances come first at the solution, which implies lower objective values.

Our crossover procedure is described by Algorithm 3 and applies the ideas proposed for BPP in [15] to JOBPRP. The algorithm receives as input a set of parents $\mathcal{G}$, the number of orders n, the weight's vector ω, and the maximum pickers' capacity c, returning the offspring $\mathcal{O}$ (a vector of individuals). The main loop of this algorithm (lines 3-30) evaluates if there are parents to be crossed. If so, at line 4, we define an empty set $\mathcal{L}$ that stores the orders which are not in any batch ($\mathcal{L}$ is

also known as free list). At line 5, we declare two individuals: $\mathcal{T}$, that we call by "tape" (which is the crude product from the parents' crossover) and $\mathcal{C}$, that is the child itself. At line 6, we choose two different parents from $\mathcal{G}$ to perform the crossover an remove them from $\mathcal{G}$ (line 7).

```
   Input  : G, n, ω, c
   Output : an offspring O
1  begin
2      O ← ∅;
3      while |G| > 0 do
4          L ← ∅;
5          T, C ← ∅, ∅;
6          G1, G2 ← random individuals from G;
7          G ← G \ {G1, G2};
8          sort G1 and G2 batches' descending by number of orders;
9          for i ← 0 to min(|G1|,|G2|) - 1 do
10             if D(G1) ≤ D(G2) then
11                 T ← T ∪ {G1[i]} ∪ {G2[i]}
12             else
13                 T ← T ∪ {G2[i]} ∪ {G1[i]}
14             end
15         end
16         if |G1| ≠ |G2| then
17             append sequentially the remaining batches to T;
18         end
19         for each g ∈ T do
20             if all g's orders are not in C then
21                 C ← C ∪ {g};
22             else
23                 append non-repeated orders from g to L;
24             end
25         end
26         for each o ∈ L do
27             insert o in some C's batch with first-fit heuristic;
28         end
29         O ← O ∪ {C};
30     end
31     return O;
32 end
```

Algorithm 3: Crossover procedure

As we are applying a controlled gene transmission technique, at each crossover we select the best genes (batches) from both parents, ensuring a high-quality child in terms of fitness. To implement this, at line 8, both $\mathcal{G}_1$ and $\mathcal{G}_2$ solutions' batches are sorted decreasingly according their number of orders (i.e., batches with more orders come first on the sorting).

Considering $\mathcal{D}$ as the picking-route distance associated to a batch (computed during the fitness evaluation from Algorithm 1), we construct $\mathcal{T}$ as follows: from line 9 to line 15 we compare batch-per-batch from $\mathcal{G}_1$ and $\mathcal{G}_2$, pushing first to $\mathcal{T}$ the batch with the lower distance, then the second lower and so on. If the parents have different sizes, we append to $\mathcal{T}$ the genes from the larger individual, which were not submitted to the previous comparison (line 17).

Note that by analyzing $\mathcal{T}$ at the end of its construction, we may verify the occurrence of repeated orders over the batches. Hence, we execute a filter step that consists of traversing $\mathcal{T}$ (line 19) while checking if all the orders from the batch $g \in \mathcal{T}$ are not in the child $\mathcal{C}$ (line 20). If this is the case, the batch is appended to the child $\mathcal{C}$ (line 21), otherwise g is not appended to $\mathcal{C}$ and the non-repeated orders from g are pushed to the free list $\mathcal{L}$ (line 23). At the end of the filtering step, we insert each order o from free list $\mathcal{L}$ in some batch of $\mathcal{C}$ using a first-fit heuristic (line 29).

V. Experiments and results

To validate our approach, we conducted several experiments of our GGACGT-JOBPRP approach over Henn's set of instances [6], provided by [12]. Also, we adapted a subset of these instances to compare our results with those provided by a commercial ILP solver running Model 1. Our implementations, complete and detailed results, instances, outputs and documentation can be found at Github [10] repository.

In the rest of this section, we present our computational environment, the datasets, the metaheuristic parameters, and analyze the summarized results.

A. Computational environment

The algorithms were implemented in `C++` language using the compiler `g++` version 9.3.0. The support scripts (instance treatment, tests automation and input/output control) were written in `Python` version 3.8.10. The tests were single-core simultaneously executed in an AMD Ryzen 9 3900X 12-Core Processor (3.8GHz) with 32 GB RAM and Ubuntu 20.04 LTS operational system. We implemented and solved Model 1 with the mathematical optimization solver Gurobi 9.0.3.

To generate pseudo-random numbers we used the Mersenne Twister algorithm. Each instance was tested 30 times, with integer seeds from [1,30] interval.

B. Instances

The Henn's set instances is the most cited dataset over warehouse's problems studies that we found. Their public files are compound by 64 instances, equally divided in four groups: ABC1, ABC2 (most requested items closer from depot) and RAN1, RAN2 (items are randomly scattered on warehouse).

Each instance have the same warehouse layout configuration: a single-block warehouse (two cross-aisles, one at bottom another on the top) with 10 aisles and 45 picking positions, resulting in 450 addresses plus depot. Each aisle have products in both right and left. Each picking position has 1 length unit (LU) and when a picker leaves or enters in an aisle it travels 1 LU more. Also, the horizontal distance from an aisle to another is 5 LU. The depot is located on the inferior cross-aisle, in front of aisle 0.

These instances are shaped combining 40, 60, 80, and 100 orders, and 30, 45, 60, and 75 as possible maximum pickers' capacities. We used this dataset as input to check the overall performance of GGACGT-JOBPRP and confront the results with literature. From the same dataset, we randomly selected one instance from each group to be modified to a smaller instance, which we truncated to the 10 first orders and fixed the capacity limit to 30. These smaller instances were used to test our formulation and assess the quality of our algorithm solutions.

C. Parameters

We used the iRace package [11] to determine POP_SIZE and PARENTS_SIZE parameters. This tool automatically detects the best parameter configuration over a parameters domain, focusing on stability and solution quality. This package provided us the values 340 for POP_SIZE and 120 for PARENTS_SIZE. The ITERATION_LIMIT parameter was obtained through controlled tests. These isolated experiments aimed to find an adequate value to this parameter so that it was possible to analyze the objective function value progression over the generations. Such value was fixed to 500. The solver execution time limit was set to 3600 s.

D. Results and analysis

To assess GGACGT-JOBPRP, we conducted experiments over the smaller instance group. The results of these experiments are shown in Table II, where the first column identifies the instance, and the second and third columns show data delivered by Gurobi and our approach, respectively. The sub-column NA represents the number of visited addresses; the sub-column LB is the solver lower bound; the sub-column OBJ is the objective value found by Gurobi with Model 1 (in the second column) and by GGACGT-JOBPRP (in the third column); TIME represents the execution time of the approaches (in seconds); GAP is the solver objective GAP, and RD is the relative difference between LB and GGACGT-JOBPRP OBJ. Our results are promising, considering the relation between the solution quality (lower than the best feasible solution provided by the solver) and the execution time, which is approximately 60 times smaller than Gurobi.

TABLE II
COMPARATIVE BETWEEN GGACGT-JOBPRP AND GUROBI

I	Gurobi					GGACGT-JOBPRP		
	NA	LB	OBJ	TIME	GAP(%)	OBJ	TIME	RD (%)
1	101	1577	1892	3600	16.64	**1794**	74.48	12.09
2	112	1832	2070	3600	11.49	**1990**	60.96	7.93
3	131	2258	2620	3600	13.81	**2500**	70.60	9.68
4	138	2478	3038	3600	18.43	**2918**	89.84	15.07

When compared with the approach of [12] (identified as MS-VNS), the GGACGT-JOBPRP demonstrated superiority in the solutions' quality. Over the 64 experiments on the dataset provided by [12], our algorithm got a better average objective value in 100% of the cases. The summary of these results can be found in Table III, where the first column identifies the instance, and the second and third columns show data from MS-VNS and GGACGT-JOBPRP approaches, respectively. RD (%) expresses the relative difference between the objective value of our approach and MS-VNS's objective value (i.e., $\frac{OBJ_{MS-VNS} - OBJ_{GGACGT-JOBPRP}}{OBJ_{MS-VNS}} \cdot 100$), while SD and CV are respectively the standard deviation and the coefficient of variation (%) from our tests. The improvement of our GGACGT-JOBPRP over the MS-VNS represents 3.1% of the average solution value and can be visualized in the graphics from figures 2 and 3. These figures illustrate the objective function contrast between the algorithms' solutions for the instances from ABC2 and RAN2 groups. Although the GGACGT-JOBPRP's average execution time (400.4 s) is more elevated than MS-VNS (43.8 s), the latter uses a faster S-Shape routing policy, avoiding solving the batches' picking-routes optimally and leading to increasing the routes' lengths.

According to Table III, our GGACGT-JOBPRP presented high stability, with small coefficient of variations ($\leq$ 1.1%, average 0.5%) over all tested seeds for each instance.

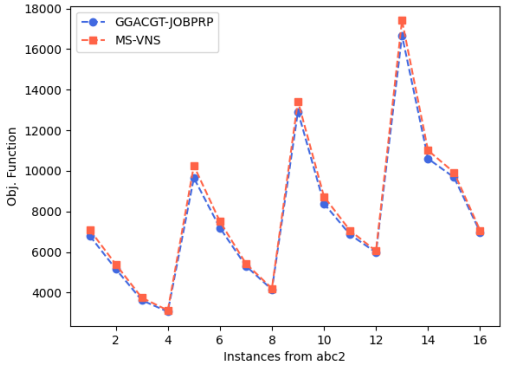

Fig. 2. ABC2 instances group - comparative with [12]

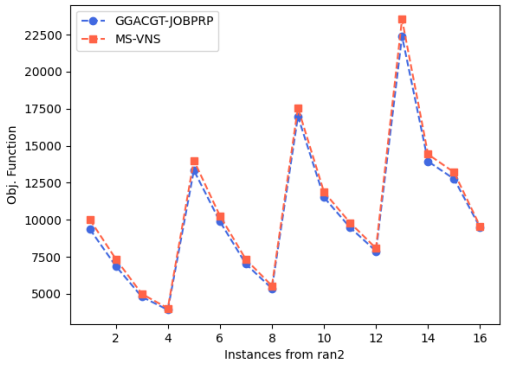

Fig. 3. RAN2 instances group - comparative with [12]

VI. CONCLUSIONS

In this work, we presented a novel ILP model and a genetic algorithm (GGACGT-JOBPRP) for the JOBPRP. The GGACGT-JOBPRP constructs a chromosome's population using a heuristic that assigns the orders to the batches, providing initial solutions with lower objective values based on the order's similarity degree. Another differential of our approach is the crossover operator that adapts controlled gene transmission, generating high-quality new individuals.

We tested our GGACGT-JOBPRP over several instances from the literature and our results were superior to the previous best known approach, considering the objective values over 100% of the experiments. We also tested our mathematical formulation within a commercial solver over a small dataset,

TABLE III
SUMMARY OF EXPERIMENTAL RESULTS

Instance	MS-VNS		GGACGT-JOBPRP				
	OBJ	TIME	OBJ	TIME	RD	SD	CV
abc1/29s-40-30	6914.0	3.4	**6542.1**	242.0	5.3	15.7	0.2
abc1/30s-40-45	4683.0	6.1	**4469.3**	183.6	4.5	29.6	0.6
abc1/31s-40-60	4255.0	9.6	**4143.4**	183.9	2.6	20.7	0.5
abc1/32s-40-75	3174.0	9.1	**3099.5**	135.9	2.3	24.0	0.7
abc1/37s-60-30	11234.0	6.9	**10667.8**	392.1	5.0	45.9	0.4
abc1/38s-60-45	7278.0	14.3	**7026.2**	312.6	3.4	32.1	0.4
abc1/39s-60-60	5727.0	34.5	**5608.1**	289.6	2.0	56.1	1.0
abc1/40s-60-75	3997.0	45.7	**3959.1**	251.8	0.9	30.1	0.7
abc1/61s-80-30	14654.0	15.2	**14169.8**	581.1	3.3	97.3	0.6
abc1/62s-80-45	10370.0	32.1	**9925.9**	489.4	4.2	34.7	0.3
abc1/63s-80-60	7301.0	57.5	**7205.0**	426.3	1.3	82.3	1.1
abc1/64s-80-75	6008.0	89.6	**5918.6**	427.3	1.4	41.3	0.6
abc1/9s-100-30	15637.0	29.7	**14794.8**	640.9	5.3	131.0	0.8
abc1/0s-100-45	10420.0	64.2	**10308.2**	631.2	1.0	51.1	0.4
abc1/1s-100-60	8449.0	116.6	**8327.0**	600.8	1.4	83.6	1.0
abc1/2s-100-75	7218.0	95.8	**7160.4**	597.2	0.7	54.7	0.7
abc2/x9l-40-30	7091.0	3.6	**6800.2**	247.2	4.1	67.8	0.9
abc2/10l-40-45	5393.0	8.4	**5163.9**	219.8	4.2	14.6	0.2
abc2/11l-40-60	3748.0	10.5	**3624.2**	166.9	3.3	30.8	0.8
abc2/12l-40-75	3128.0	13.2	**3061.6**	147.9	2.1	19.5	0.6
abc2/17l-60-30	10250.0	7.6	**9651.5**	376.1	5.8	64.6	0.6
abc2/18l-60-45	7509.0	16.7	**7195.0**	322.9	4.1	39.0	0.5
abc2/19l-60-60	5431.0	36.5	**5313.0**	287.6	2.1	42.2	0.7
abc2/20l-60-75	4207.0	33.6	**4151.1**	256.2	1.3	27.6	0.6
abc2/45l-80-30	13428.0	14.6	**12920.4**	520.5	3.7	85.1	0.6
abc2/46l-80-45	8735.0	33.5	**8388.6**	449.2	3.9	45.9	0.5
abc2/47l-80-60	7062.0	52.7	**6864.8**	436.4	2.7	46.9	0.6
abc2/48l-80-75	6048.0	93.2	**5977.3**	421.9	1.1	40.0	0.6
abc2/3l-100-30	17408.0	24.4	**16668.3**	682.7	4.2	93.6	0.5
abc2/4l-100-45	11005.0	65.4	**10602.2**	618.1	3.6	61.5	0.5
abc2/5l-100-60	9920.0	101.2	**9676.0**	638.2	2.4	68.7	0.7
abc2/6l-100-75	7069.0	86.8	**6984.5**	599.6	1.1	50.7	0.7
ran1/29s-40-30	9569.0	3.3	**9099.8**	256.6	4.9	40.1	0.4
ran1/30s-40-45	6243.0	6.6	**5901.4**	181.2	5.4	22.3	0.3
ran1/31s-40-60	5631.0	8.8	**5400.4**	163.6	2.2	20.8	0.3
ran1/32s-40-75	4385.0	26.0	**4289.0**	155.5	1.1	40.7	0.9
ran1/37s-60-30	14960.0	7.6	**14417.0**	406.8	3.6	94.2	0.6
ran1/38s-60-45	9626.0	29.5	**9190.0**	312.1	4.4	109.9	1.1
ran1/39s-60-60	7760.0	48.2	**7453.5**	299.2	3.9	33.3	0.4
ran1/40s-60-75	5473.0	24.9	**5334.8**	241.5	2.5	36.6	0.6
ran1/61s-80-30	19677.0	17.2	**18654.0**	597.0	5.1	74.9	0.4
ran1/62s-80-45	14003.0	49.8	**13470.0**	479.0	3.8	122.8	0.9
ran1/63s-80-60	9784.0	102.0	**9487.4**	421.7	2.0	40.7	0.4
ran1/64s-80-75	8010.0	67.1	**7763.7**	411.1	1.0	41.8	0.5
ran1/9s-100-30	21127.0	29.6	**20285.0**	646.6	3.9	160.4	0.7
ran1/0s-100-45	14425.0	64.1	**13929.8**	631.6	3.4	65.7	0.4
ran1/1s-100-60	11417.0	167.6	**11230.0**	620.7	1.6	52.9	0.4
ran1/2s-100-75	9395.0	108.0	**9218.8**	600.7	1.8	50.7	0.5
ran2/x9l-40-30	10020.0	3.5	**9372.8**	256.7	6.4	76.7	0.8
ran2/10l-40-45	7357.0	7.1	**6877.8**	210.7	6.5	36.9	0.5
ran2/11l-40-60	4998.0	10.7	**4799.8**	155.3	3.9	24.2	0.5
ran2/12l-40-75	4028.0	15.7	**3921.8**	138.1	2.6	22.4	0.5
ran2/17l-60-30	14009.0	9.4	**13338.5**	393.3	4.7	106.3	0.7
ran2/18l-60-45	10275.0	28.9	**9890.8**	337.2	3.7	57.7	0.5
ran2/19l-60-60	7323.0	32.0	**7013.4**	277.9	4.2	46.4	0.6
ran2/20l-60-75	5541.0	24.7	**5359.4**	241.6	3.2	39.7	0.7
ran2/45l-80-30	17558.0	17.7	**16975.0**	507.1	3.3	55.9	0.3
ran2/46l-80-45	11879.0	67.1	**11516.3**	469.6	3.0	48.0	0.4
ran2/47l-80-60	9806.0	47.8	**9484.5**	425.8	3.2	49.8	0.5
ran2/48l-80-75	8076.0	87.9	**7883.2**	410.7	2.3	30.3	0.3
ran2/3l-100-30	23532.0	31.9	**22388.7**	693.2	4.8	152.9	0.6
ran2/4l-100-45	14423.0	94.4	**13944.4**	652.8	3.3	81.2	0.5
ran2/5l-100-60	13203.0	93.8	**12756.0**	639.0	3.3	44.3	0.3
ran2/6l-100-75	9553.0	238.0	**9479.8**	616.6	0.7	106.7	1.1
Average	9340.8	43.7	**9007.7**	400.4	3.1	56.4	0.5

verifying the quality of our metaheuristic solutions for these smaller instances.

As future work, we intend to improve our mathematical model and present a new matheuristic to JOBPRP, hybridizing the crossover operator from GGACGT-JOBPRP. Also, we are planning to extend the operations for multi-block warehouses. One of the major difficulties encountered during this work was to find clear datasets accompanied by granular and specific results, thus, we also plan to construct and share new detailed datasets to be used as instances in warehouse problems.

REFERENCES

[1] Babiche Aerts, Trijntje Cornelissens, and Kenneth Sörensen. The joint order batching and picker routing problem: Modelled and solved as a clustered vehicle routing problem. *Computers & Operations Research*, 129:105168, 2021.

[2] Olivier Briant, Hadrien Cambazard, Diego Cattaruzza, Nicolas Catusse, Anne-Laure Ladier, and Maxime Ogier. An efficient and general approach for the joint order batching and picker routing problem. *European Journal of Operational Research*, 285(2):497–512, 2020.

[3] Statista Research Department. Retail e-commerce revenue in the united states from 2017 to 2025. https://www.statista.com/statistics/272391/us-retail-e-commerce-sales-forecast/, January 2022. Accessed: 2022-01-04.

[4] Emanuel Falkenauer. A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2:5–30, 1996.

[5] NOUD GADEMANN and VAN DE STEEF VELDE. Order batching to minimize total travel time in a parallel-aisle warehouse. *IIE Transactions*, 37(1):63–75, 2005.

[6] Sebastian Henn and Gerhard Wäscher. Tabu search heuristics for the order batching problem in manual order picking systems. *European Journal of Operational Research*, 222(3):484–494, 2012.

[7] Sarah Hofstetterl. 2021 e-commerce year in review. https://www.forbes.com/sites/sarahhofstetter/2021/12/27/2021-e-commerce-year-in-review/, December 2021. Accessed: 2022-01-04.

[8] Sören Koch and Gerhard Wäscher. A grouping genetic algorithm for the order batching problem in distribution warehouses. *Journal of Business Economics volume*, 86:131–153, 2016.

[9] Osman Kulak, Yusuf Şahin, and Mustafa Taner. Joint order batching and picker routing in single and multiple-cross-aisle warehouses using cluster-based tabu search algorithms. *Flexible Services and Manufacturing Journal - FLEX SERV MANUF J*, 24, 03 2012.

[10] Felipe Furtado Lorenci and Santiago Valdés Ravelo. Jobprp implementation. https://github.com/JOBPRP/jobprp_implementation, February 2022. Accessed: 2022-02-13.

[11] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[12] Borja Menéndez, Eduardo G. Pardo, Antonio Alonso-Ayuso, Elisenda Molina, and Abraham Duarte. Variable neighborhood search strategies for the order batching problem. *Computers & Operations Research*, 78:500–512, 2017.

[13] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329, oct 1960.

[14] Lucie Pansart, Nicolas Catusse, and Hadrien Cambazard. Exact algorithms for the order picking problem. *Computers & Operations Research*, 100:117–127, 2018.

[15] Marcela Quiroz-Castellanos, Laura Cruz-Reyes, Jose Torres-Jimenez, Claudia Gómez S., Héctor J. Fraire Huacuja, and Adriana C.F. Alvim. A grouping genetic algorithm with controlled gene transmission for the bin packing problem. *Computers & Operations Research*, 55:52–64, 2015.

[16] H. Ratliff and Arnon Rosenthal. Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem. *Operations Research*, 31:507–521, 06 1983.

[17] A. Scholz and G. Wäscher. Order Batching and Picker Routing in manual order picking systems: the benefits of integrated routing. *Central European Journal of Operations Research*, 25:491–520, 2017.