Computers & Industrial Engineering 182 (2023) 109388

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

A Memetic Algorithm for the multi-product Production Routing Problem

Luiz Fernando Rodrigues [a,*], Maristela Oliveira Dos Santos [a], Bernardo Almada-Lobo [b]

[a] Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, Brazil
[b] INESC TEC and Faculty of Engineering, University of Porto, Porto, Portugal

ARTICLE INFO

Keywords:
Production Routing Problem
Memetic algorithms
Mixed integer optimization

ABSTRACT

This article addresses the Production Routing Problem (PRP), which consists of determining, in an integrated way, production and inventory planning, and vehicle routing to minimize the costs involved. In the problem, a plant is responsible for producing several types of products to meet the known demand of a set of customers using a homogeneous fleet of vehicles over the planning horizon. In the literature, evolutionary approaches have not been explored in depth for the PRP, specifically for the problem with multiple products. Thus, this work mitigates this gap, presenting a novel Memetic Algorithm and testing its effectiveness on randomly generated sets of instances, comparing the results obtained with a commercial optimization solver. In our solution approach, several classic operators from the literature were implemented. Furthermore, we propose four novel genetic operators. In addition, we evaluated the proposed method's performance in classical instances of literature considering a single item. The computational experiments were carried out to assess the impact of the numerous parameter combinations involving the metaheuristic, and, from statistical analyses, we evidence the proposed technique's robustness. Computational experiments showed that our proposed method outperforms the commercial solver Gurobi in determining feasibly high-quality solutions, mainly on large instances for the PRP with multiple items.

1. Introduction

Problems involving production, inventory, and distribution planning have been widely studied in recent decades. However, publications considering these decisions integrated at different levels are still scarce in the literature (Absi et al., 2018; Adulyasak et al., 2015; Moons et al., 2017). The Production Routing Problem (PRP) addresses two important aspects of supply chain planning. The first is the lot-sizing problem (LSP) which involves aspects inherent to production and inventory planning, such as determining how much and when to produce to meet the demand, while minimizing inventory, preparation, and production costs. The second is the vehicle routing problem (VRP), which determines how vehicle fleets will be used to deliver products, meet customer demands and minimize distribution costs. Traditionally, these problems were optimized sequentially, i.e., production and inventory planning served as input parameters for defining delivery routes or vice versa. Although the complexity of the integrated problem is greater, this strategy provides an important and significant reduction in production chain costs (Adulyasak et al., 2015). Chandra and Fisher (1994) showed that an integrated approach could generate savings costs of up to 20% when compared to the sequential approach.

In the PRP, we must decide, at a plant, the production of multiple items over a finite planning horizon discretized into periods. The production capacity is limited, and the distribution of products to customers is carried out using a homogeneous and limited fleet of vehicles. Products can be stored at the plant and with customers with different inventory costs. In summary, the main objective of the problem is to minimize production costs, plant and customer inventory costs, and routing costs, respecting the capacity constraints to meet customer demand.

The PRP is also known as the Production, Inventory, Distribution, and Routing Problem (PIDRP) or as the Integrated Production and Distribution Problem (IPDP), as highlighted by Miranda, Cordeau et al. (2018). Some reviews of this problem can be found in Darvish et al. (2021), Absi et al. (2018), Moons et al. (2017), Adulyasak et al. (2015) and Mostafa and Eltawil (2015).

In the problem, we consider the plant or supplier to be responsible for managing inventories held by customers, an integration policy called VMI (*vendor managed inventory*). Therefore, the plant will decide when and how customer inventories will be replenished, ensuring that demand is met throughout the planning horizon. Regarding inventories, the maximum inventory level (ML) policy is adopted, thus allowing the quantities delivered to each customer to have any positive value, as long as the maximum inventory level is not exceeded. Archetti et al.

* Corresponding author.
E-mail address: luiz.rodrigues@uftm.edu.br (L.F. Rodrigues).

https://doi.org/10.1016/j.cie.2023.109388
Received 25 September 2022; Received in revised form 11 May 2023; Accepted 12 June 2023
Available online 20 June 2023

(2011) carry out a comparative study between the ML and order-up-to-level (OU) policy, analyzing the costs and characteristics of solutions in different contexts. The OU policy establishes that the quantities delivered to each customer should always raise the inventory level to the maximum allowed value.

This paper proposes a Memetic Algorithm (MA) for the multi-product PRP. This paper's main contribution is using a population technique that is still little explored in the PRP context. Although some research uses these techniques, they are for other approaches to the problem, such as the pioneering work of Boudia and Prins (2009), which proposes a memetic algorithm with population management for the PRP with a single item. A hybrid genetic algorithm was also developed by Izadi et al. (2020) and a memetic algorithm by Yagmur and Kesen (2021) for the PRP considering scheduling. This paper, however, aims to develop a memetic algorithm using a hierarchical population structure to organize individuals and improve overall method performance.

Since the method is a robust population evolutionary metaheuristic involving the adjustment of many parameters with the incorporation of strategies developed specifically for the problem, its application in this context becomes challenging. In this regard, to generate a set of solutions that would form the initial population, we divided the original problem into two subproblems using classic strategies to solve each one separately but in a way that, at the end of the process, solves the integrated problem. Furthermore, we modified the algorithms to enable the generation of the necessary number of solutions to compose the initial population. We also developed four new one-point and two-point genetic operators for the integrated problem and adapted another classic VRP operator. Several strategies were adapted and used in the local search as part of the process of intensifying the search for better solutions. The method was evaluated considering a set of randomly generated instances, and the results were compared with an optimization solver. Finally, we compared the technique's performance with a state-of-the-art approach involving the PRP with a single item. It is important to note that the proposed MA performed much better than the solver in instances with multiple items generated in this article. Although the entire development of this paper focused on the problem with multiple items, the proposed technique presented good performance compared to the state-of-the-art approach involving the problem with a single item.

Thus, we can list the main contributions of this article as follows:

(i) we developed an MA with a hierarchical and structured population for the multi-product PRP and a homogeneous fleet, but which can also be used for the classic problem with only one product or multi-product PRP with the heterogeneous fleet;
(ii) we developed three one-point and one two-point crossover operators and four novel mutation operators designed explicitly for the PRP;
(iii) we present decomposition strategies, from the adaptation of classic LSP and VRP strategies, with excellent computational performance to obtain a numerous and diverse set of feasible solutions for the integrated problem;
(iv) we developed an improvement heuristic that combines several classic strategies from the literature capable of efficiently exploring the solution space, and that also, due to its efficiency and implementation simplicity, can be used in real problems involving large instances;
(v) we generated sets of instances for the execution of computational experiments. Data will be available to serve as a database for future comparisons and as an extension of the proposed methods.

This article is organized as follows: Section 2 presents a literature review with the most recent and relevant published articles. The Section 3 defines the mathematical model with the characteristics of the problem addressed herein. In Section 4, we present the components of the proposed memetic algorithm with the computational aspects of its implementation. In Section 5 the computational results are reported and analyzed statistically and, finally, in Section 6 the conclusions and some perspectives for future research are presented.

2. Literature review

This section briefly reviews articles that used exact methods and considers the PRP with a single item. Afterward, we address those works that consider the multi-product problem, which is the focus of this research. Finally, we review research that used genetic and memetic algorithms in the context of the PRP or its extensions.

2.1. Classic production routing problem

PRP is an NP-hard problem (Adulyasak et al., 2015; Archetti et al., 2011; Boudia et al., 2007), so exact methods have their application restricted to small and medium-sized instances. Considering the PRP with a single plant, one product, and the use of the branch-and-cut technique for its resolution, we cite Ruokokoski et al. (2010) for the problem with only one incapacitated vehicle and Qiu, Wang et al. (2018), which considers multiple capacitated homogeneous vehicles with time windows. By studying real problems involving perishable products, Qiu et al. (2019) analyzed the impact of inventory supply policies. Considering the growing concern with ecological operations and the impact of short-term decisions on carbon emissions, Darvish et al. (2019) propose a mixed integer optimization model. In turn, Ramos et al. (2020) proposed a formulation based on network flow that was developed considering multiple trips with time windows, and Zhang et al. (2021) developed a technique using Benders decomposition.

Most articles considering the problem with a single plant and product use the instances proposed by Archetti et al. (2011) and Boudia et al. (2007), combining exact and heuristic methods. Among them, we highlight Adulyasak et al. (2014), who propose a method that uses an adaptative large neighborhood search algorithm (ALNS) to handle the binary variables while the continuous variables are optimized using a network flow model. We also highlight Absi et al. (2015), who presents an iterative heuristic to solve the lot-sizing problem and routing problem separately. A decomposition heuristic that uses exact methods to solve the problem in steps is proposed by Solyalı and Süral (2017), while Russell (2017) presents techniques based on mathematical programming combined with a metaheuristic and an iterative procedure.

Qiu et al. (2018a) present a metaheuristic based on variable neighborhood search (VNS) and variable neighborhood descent (VND) modified and combined with exact methods. Considering a visit spacing policy, Avci and Yildiz (2019) present two mixed integer models that are solved using an iterative algorithm based on programming mathematics. Chitsaz et al. (2019) propose a general model for the assembly routing problem (ARP), implementing a decomposition technique combined with exact methods. A new formulation based on two-commodity flow and a two-phase algorithm that explores the infeasible solution space are proposed by Manousakis et al. (2022). Vadseth et al. (2022) present a path-flow formulation and develop an improvement heuristic with multiple restarts combined with exact methods.

2.2. Multi-product production routing problem

Chandra and Fisher (1994) were pioneers in introducing the integrated production and routing problem involving multiple products and a single plant, considering the distribution of products to a set of customers using a fleet of homogeneous trucks. Two techniques were proposed, one that decomposes and solves the LSP and the VRP separately and another that solves the problem in an integrated way.

Table 1
Classification of the papers on the Multi-Product PRP with single plant.

Papers	Inventory	Routing		Solution method	Instances		Extension
	Policy	Fleet	#Vehicles	Approach	Classic	Real	
Chandra and Fisher (1994)	ML	Hom.	Unlimited	Decomposition			
Fumero and Vercellis (1999)	ML	Hom.	Limited	Lagrangian relaxation			
Armentano et al. (2011)	ML	Hom.	Unlimited	Tabu search	B		
Brahimi and Aouam (2016)	ML	Hom.	Single	Relax-and-fix			Backordering
Miranda, Cordeau et al. (2018)	ML	Het.	Limited	Decomposition		Yes	Multiple time windows
Miranda, Morabito et al. (2018)		Hom.	Single	Relax-and-fix		Yes	Multiple time windows
Qiu et al. (2018b)		Hom.	Limited	Branch-and-cut		Yes	Startup cost
Neves-Moreira et al. (2019)	ML	Het.	Limited	Fix-and-optimize	A	Yes	Time windows
Mostafa and Eltawil (2019)		Het.	Limited	Iterative/Branch-and-cut	A1		
Li et al. (2019)	ML	Hom.	Limited	Fix-and-optimize	A, B		Outsourcing
Salehi Sarbijan and Behnamian (2020)		Hom.	Limited	Particle swarm optimization			Outsourcing and carbon emission
This paper	ML	Hom.	Limited	Memetic algorithm	A		

| Hom. - Homogeneous | Het. - Heterogeneous | A - Archetti et al. (2011) | B - Boudia et al. (2007) |

Considering a limited fleet of homogeneous vehicles, Fumero and Vercellis (1999) propose a technique that solves the capacitated LSP and the VRP separately, using Lagrangian relaxation to separate the transport and production variables, but preserving the global optimization perspective.

For the problem with a single plant and using an unlimited and homogeneous fleet of vehicles, Armentano et al. (2011) present two solution methods, one using Tabu Search and the other Tabu Search combined with path relinking. Brahimi and Aouam (2016) formulate two new models in which customer demands can be delayed, but with the insertion of penalties in the objective function. The authors use a relax-and-fix heuristic to solve a modified version of the LSP combined with a local search to construct the delivery routes and, thus, find a complete solution to the integrated problem.

Upon incorporating realistic features into the model, Miranda, Cordeau et al. (2018) approach the PRP in the context of a furniture plant and implement a decomposition heuristic to solve the problem iteratively in two steps. Likewise, Miranda, Morabito et al. (2018) present a model in which a furniture plant produces items used in the assembly of final products and which will be delivered to customers by a single vehicle. The authors propose six relax-and-fix heuristics, which explore different criteria to partition and fix the variables, and two hybrid heuristics in which the initial solution is built and then optimized by a commercial solver.

Qiu et al. (2018b) in turn formulate a mixed integer model for the PRP considering start-up costs and a homogeneous fleet of vehicles. Furthermore, the authors implement a branch-and-cut technique using three families of valid inequalities to more tightly fit the proposed model. Inspired by a real problem involving a single meat processing center with several production lines, Neves-Moreira et al. (2019) propose a technique that combines fix-and-optimize methods to solve the problem in three phases.

In order to reduce the complexity of the problem, Mostafa and Eltawil (2019) introduce valid inequalities for the PRP in regard to a heterogeneous fleet of vehicles. Li et al. (2019) formulate a model for the problem considering that production and delivery can be outsourced, implementing a three-level mathematical programming-based heuristic. When considering the effects of outsourcing and greenhouse gas emissions, Salehi Sarbijan and Behnamian (2020) address the green PRP and propose a resolution strategy using PSO (particle swarm optimization).

2.3. Genetic and memetic algorithms applied to PRP

Faced with challenges in terms of computational complexity of the problem, a promising alternative is the use of metaheuristics, especially for large problems with more realistic characteristics. Briefly, we highlight some research that used genetic and memetic algorithms in the context of the PRP or its extensions.

Boudia and Prins (2009) developed a memetic algorithm with different population management strategies for the PRP with a single plant and a single item. In regards to the PRP together with production scheduling, we have Izadi et al. (2020), who used the optimal solution's dominance properties as part of a genetic algorithm's hybridization process. There is also Yagmur and Kesen (2021), who proposes a memetic algorithm and simulated annealing as solution methods. When considering the external depot production routing problem, which is an extension of the classic PRP, Kayé et al. (2021) developed a modified memetic algorithm in which the mutation is replaced by three local searches that are applied in vehicle routing.

Table 1 summarizes the research that considers the multi-product PRP and a single plant. It is important to note that all articles propose new instances for the multi-product PRP, but only 5 use the classic PRP instances with a single item for comparison. Among these works of research, 4 use instance set A, 1 uses set B, and 1 used both sets. In addition, 4 articles present instances involving real problems and 7 present models that consider additional characteristics, with emphasis on time windows and outsourcing. Finally, it should be noted that among the resolution techniques, it is only this article that used memetic algorithms. We present below the mathematical model proposed for the problem herein addressed.

3. Mathematical modeling

The mathematical model is an adaptation of work carried out by Armentano et al. (2011), Fumero and Vercellis (1999). In this article, we consider a plant with limited production and storage capacity, producing a set of products that will be distributed to a set of customers in order to meet their respective demands in a finite planning horizon. The products can be stored at the plant or held by the customers with inventory management being performed by the supplier (VMI). Furthermore, we adopted the same maximum level (ML) policy hypotheses, considered by Fumero and Vercellis (1999), allowing the quantities delivered to each customer to have any positive value as long as the maximum inventory level is not exceeded. The product distribution is carried out by a fleet of homogeneous vehicles with limited capacity, considering the fixed and variable costs for the use of each vehicle. A customer can be visited by only one vehicle in each period, and the vehicle can only make one trip per period, leaving and returning to the plant at the end of said period. The objective of the problem is to minimize production and setup costs, plant and customer inventory costs, in addition to vehicle routing costs. Next, we define the form of representation, the parameters, and the decision variables of the model.

Consider a complete graph $G = (W, E)$, where $W = \{0, 1, \ldots, N\}$ is a set of nodes representing the plant and the customers and, $E = \{(i,k) : i,k \in W, i \neq k\}$ is the set of edges, which are the routes between the plant and the customers. The plant, indicated by node

$i = 0$, manages customer inventories $i = 1, \ldots, N$ in a finite planning horizon, $t = 1, \ldots, T$. The delivery of products will be made by a limited homogeneous fleet of vehicles $v = 1, \ldots, V$ from the supplier. We present below a summary of these parameters and decision variables:

Parameters:

B = Production capacity;

b_p = Time required to produce item p;

c_p = Production cost of item p;

s_p = Setup cost of item p;

M = Big number (for example: $\sum_p \sum_i \sum_t d_{pit}$);

U_{pi} = Maximum inventory upper bound of item p at site i;

I_{pi0} = Initial Inventory of item p at site i;

h_{pi} = Inventory cost of item p at site i;

C = Vehicle capacity;

f = Fixed transportation cost;

a_{ik} = Transportation cost for traveling from node i to node k;

d_{pit} = Demand of item p at customer i in period t.

Decision variables:

x_{pt} = Quantity of item p produced in period t.

$$y_{pt} = \begin{cases} 1, & \text{if item } p \text{ is produced in period } t; \\ 0, & \text{otherwise.} \end{cases}$$

I_{pit} = Inventory of item p at site i in the end of period t.

$$z_{vikt} = \begin{cases} 1, & \text{if vehicle } v \text{ travels along edge } (i,k) \text{ in period } t; \\ 0, & \text{otherwise.} \end{cases}$$

r_{pvikt} = Quantity of item p transported by vehicle v on edge (i,k) in period t;

q_{pvit} = Quantity of item p delivered by vehicle v to customer i in period t.

We assume that the quantities produced in period t will be ready for delivery in the same period. Thus, we have the following general PRP model:

Minimize

$$\sum_{p=1}^{P}\sum_{t=1}^{T}(s_p y_{pt} + c_p x_{pt}) + \sum_{p=1}^{P}\sum_{i=0}^{N}\sum_{t=1}^{T} h_{pi} I_{pit} + \sum_{v=1}^{V}\sum_{k=1}^{N}\sum_{t=1}^{T} f z_{v0kt} + \sum_{v=1}^{V}\sum_{\substack{i,k=0 \\ i\neq k}}^{N}\sum_{t=1}^{T} a_{ik} z_{vikt} \tag{1}$$

Subject to:

$$x_{pt} + I_{p0,t-1} - \sum_{v=1}^{V}\sum_{i=1}^{N} q_{pvit} = I_{p0t} \quad 1 \le p \le P \quad 1 \le t \le T \tag{2}$$

$$\sum_{v=1}^{V} q_{pvit} + I_{pi,t-1} - d_{pit} = I_{pit} \quad 1 \le p \le P \quad 1 \le i \le N \quad 1 \le t \le T \tag{3}$$

$$\sum_{p=1}^{P} b_p x_{pt} \le B \quad 1 \le t \le T \tag{4}$$

$$x_{pt} \le M y_{pt} \quad 1 \le p \le P \quad 1 \le t \le T \tag{5}$$

$$I_{pit} \le U_{pi} \quad 1 \le p \le P \quad 0 \le i \le N \quad 1 \le t \le T \tag{6}$$

$$\sum_{\substack{i=0 \\ i\neq k}}^{N} r_{pvikt} - \sum_{\substack{l=0 \\ l\neq k}}^{N} r_{pvklt} = q_{pvkt} \quad 1 \le p \le P \quad 1 \le v \le V \quad 1 \le k \le N \quad 1 \le t \le T \tag{7}$$

$$\sum_{v=1}^{V}\sum_{k=1}^{N} r_{pv0kt} - \sum_{v=1}^{V}\sum_{i=1}^{N} r_{pvi0t} = \sum_{v=1}^{V}\sum_{l=1}^{N} q_{pvlt} \quad 1 \le p \le P \quad 1 \le t \le T \tag{8}$$

$$\sum_{p=1}^{P} r_{pvikt} \le C z_{vikt} \quad 1 \le v \le V \quad 0 \le i \le N \quad 0 \le k \le N \quad 1 \le t \le T \quad i \neq k \tag{9}$$

$$\sum_{k=1}^{N} z_{v0kt} \le 1 \quad 1 \le v \le V \quad 1 \le t \le T \tag{10}$$

$$\sum_{\substack{i=0 \\ i\neq k}}^{N} z_{vikt} - \sum_{\substack{l=0 \\ l\neq k}}^{N} z_{vklt} = 0 \quad 1 \le v \le V \quad 0 \le k \le N \quad 1 \le t \le T \tag{11}$$

$$\sum_{v=1}^{V}\sum_{\substack{i=0 \\ i\neq k}}^{N} z_{vikt} \le 1 \quad 1 \le k \le N \quad 1 \le t \le T \tag{12}$$

$$y_{pt}, z_{vikt} \in \{0,1\} \quad 1 \le p \le P \quad 1 \le v \le V \quad 0 \le i \le N \quad 0 \le k \le N \quad 1 \le t \le T \quad i \neq k \tag{13}$$

$$x_{pt}, I_{pit}, r_{pvikt}, q_{pvit} \ge 0 \quad 1 \le p \le P \quad 1 \le v \le V \quad 0 \le i \le N \quad 0 \le k \le N \quad 1 \le t \le T \quad i \neq k \tag{14}$$

The objective function (1) minimizes the inventory, production, and distribution costs, with the first term representing the item preparation and production costs. Regarding the second term, we have the inventory costs of the products at the plant and held by the customers. Finally, the fixed costs for using each vehicle and the variable costs for traversing the routes during item distribution to the customers are computed. The constraints in (2) establish the inventory balance at the plant according to production and the quantities sent to customers. In the constraints in (3), there is an inventory balancing of the inventory held by customers, considering the deliveries received and the meeting of demands. Fumero and Vercellis (1999) point out that if the products were not stored with the customers, then, in the constraints in (3), we would have $\sum q = d$. Hence, demands would be used in the constraints in (2) and, consequently, production and distribution problems could be treated separately. The constraints in (4) define the plant's maximum production capacity in each period. The constraints in (5) establish the relationship between production and preparation variables, i.e., if there is no setup, then there can be no item production in this period. The constraints in (6) delimit the maximum inventory capacity for items at the plant and those held by customers. The constraints in (7) and (8) are the equations for conserving the flow of items, ensuring balance at each customer location and collection at the plant. According to Fumero and Vercellis (1999), the constraints in (7) were defined separately for each vehicle to prevent transshipment at intermediate nodes, i.e., these constraints together with the non-negativity of the variable q_{pvit} prevent a vehicle from transporting items between customers. Furthermore, the constraints in (7) and (8) also guarantee the elimination of subroutes. The maximum vehicle load capacity is delimited by the constraints in (9). Fumero and Vercellis (1999) emphasize that it would suffice to verify vehicle capacity at the plant ($i = 0$), since the vehicles do not collect products from the customers. However, constraints are maintained for all locations in order to ensure the relationship between binary variables (z_{vikt}) and transport variables (r_{pvikt}). The constraints in (10) impose at most one route for each vehicle per period and the

constraints in (11) ensure that routes can only end at the plant, thus guaranteeing the return of vehicles to the place of departure at the end of each period. The constraints in (12) guarantee that one vehicle at most can visit a customer in each period, thus avoiding fractional deliveries. Lastly, the constraints in (13) and (14) define the variable types.

In order to obtain a more general model, we chose not to adopt a minimum safety level for inventories, as done (Fumero & Vercellis, 1999). Furthermore, we did not establish the maximum route size, assuming that all vehicles in the fleet are capable of traversing any of the possible routes, but they are naturally limited by load capacities.

4. Memetic algorithm

Genetic algorithms, proposed by John Holland in the mid-seventies, emerged from the analogy between natural selection and genetic mechanisms and from search techniques to obtain optimization problem solutions.

In a genetic algorithm, a set of individuals, called the initial population, is randomly generated. Each individual is assigned a fitness value, in general, related to the objective function value, and therefore, each individual represents a point in the problem's search space. The reproduction, crossover, and mutation operators are then applied to pairs of individuals (parents) that were chosen through a selection mechanism, giving rise to new individuals (children). The process's repetitions will produce a new set of individuals that will form a new population, initially more evolved, considering that it is an evolution model that incorporates survival concepts and selection of the most adapted.

By incorporating local search strategies into genetic algorithms, we have the so-called memetic algorithms (Moscato & Norman, 1992) or hybrid genetic algorithms. In memetic algorithms, individuals undergo a process of cultural evolution that occurs through a local search applied to the children after the execution of the reproduction operators. In this case, individuals are called agents and have problem-specific information that was incorporated by the applied local search.

Fig. 1 presents the elementary operating structure of this metaheuristic in a flowchart.

In the following sections, we describe the basic elements that define the proposed memetic algorithm, as well as relevant and necessary aspects of its implementation required to find a good PRP solution.

4.1. Solution representation

We used a complete representation of the solution to the problem, i.e., an individual will store the quantities produced, the inventory, and the information regarding the routes traveled by the vehicles. In Fig. 2 we have a graphical representation of this structure.

4.2. Fitness evaluation

The evaluation function stores the cost and a measure related to each individual's level of infeasibility. Thus, we establish different levels of infeasibility according to the number of constraints that are being violated, i.e., an integer indicating the types of unmet constraints, such as capacity constraints, factory or customer inventory, vehicle capacity, and so on. Thus, we define the following evaluation function

$$f(ind) = (cost, \mu)$$

Being:

ind = individual;

$cost$ = value of the solution in the objective function;

μ = number of constraints violated.

Therefore, the fitness value will be defined by an ordered pair that stores the cost and the number of constraints violated by this individual. Obviously, if the individual is feasible, then $\mu = 0$.

4.3. Population structure

In order to organize the individuals, we adopted a tree-structured population. This structure was initially proposed by Berretta and Moscato (1999) for the number partitioning problem and was also used in other production planning problems (França et al. (2001), Mendes et al. (2002), Berretta and Rodrigues (2004), Toledo et al. (2009)). The model consists of a ternary tree with three levels and a total of 26 individuals, as shown in Fig. 3.

The population is hierarchically organized by leaders and subordinates, classified according to the value assigned by the evaluation function. We, therefore, have a structure ordered according to the quality of the solutions that these individuals represent for the problem. By analyzing each level of the tree, we can divide the structure into four sub-populations. Thus, agent 1 will be the leader of the sub-population that has agents 2, 3 and 4 as subordinates. Agent 2 will be the leader of the sub-population that has agents 5, 6 and 7 as subordinates, and so on. Each agent stores two individuals, one called a pocket and the other called a current, totaling 26 individuals in the population. The pocket will always have a better value than the current and when this does not occur, individuals switch positions. A leader's pocket will have a better value than its subordinates. Whenever modifications are performed on individuals, a hierarchical update procedure will be performed, thus guaranteeing the verification between pockets and currents and also between leaders and subordinates. Thus, the best individual from the population will always be stored in agent 1's pocket individual.

4.4. Initial population

In order to generate an initial set of solutions to compose the initial population, we used a combination of strategies widely implemented to separately obtain solutions to the routing and lot-sizing problems.

Thus, the process of obtaining a solution for the PRP was divided into two steps. In the first step, a heuristic was used to determine a solution to the vehicle routing problem, guaranteeing that customer demands were met. With the definition of deliveries, the customer's inventories are updated. In the second step, a heuristic determines the solution to the lot-sizing problem using the demands established in the first step. Finally, by incorporating the routing decisions with the production plan, we have a solution to the integrated problem and, therefore, an individual for the initial population. It should be noted that the order of these steps can be inverted.

4.4.1. Initial solution: routing problem

Next, we detail the heuristics implemented to determine the solutions to the vehicle routing problem.

(i) **BFD**: The BFD heuristic (best fit decreasing) solves a binary bin packing problem in each period, assigning loads to vehicles in order to meet customer demands. Customer orders are loaded onto vehicles, from the largest to the smallest, respecting load capacities. If the number of vehicles in the fleet is insufficient to meet the demand for the period, then an adjustment procedure is executed. This procedure carries out attempts to load the remaining lots in previous periods, prioritizing vehicles with greater unused capacity.

(ii) **BFD-Inverse**: Aiming to generate several individuals for the initial population, we implemented a second version of the BFD heuristic, inverting the order of loading the orders into the vehicles, i.e., the orders will be loaded from smallest to largest. As with the BFD, if the demand is not met for some period, an adjustment procedure is executed.

(iii) **BFD-Rand**: Finally, in the third BFD heuristic variant, the loading order will be random, i.e., it will be obtained by randomly drawing the orders in each period. Analogous to previous versions, the adjustment procedure, if necessary, will be applied.

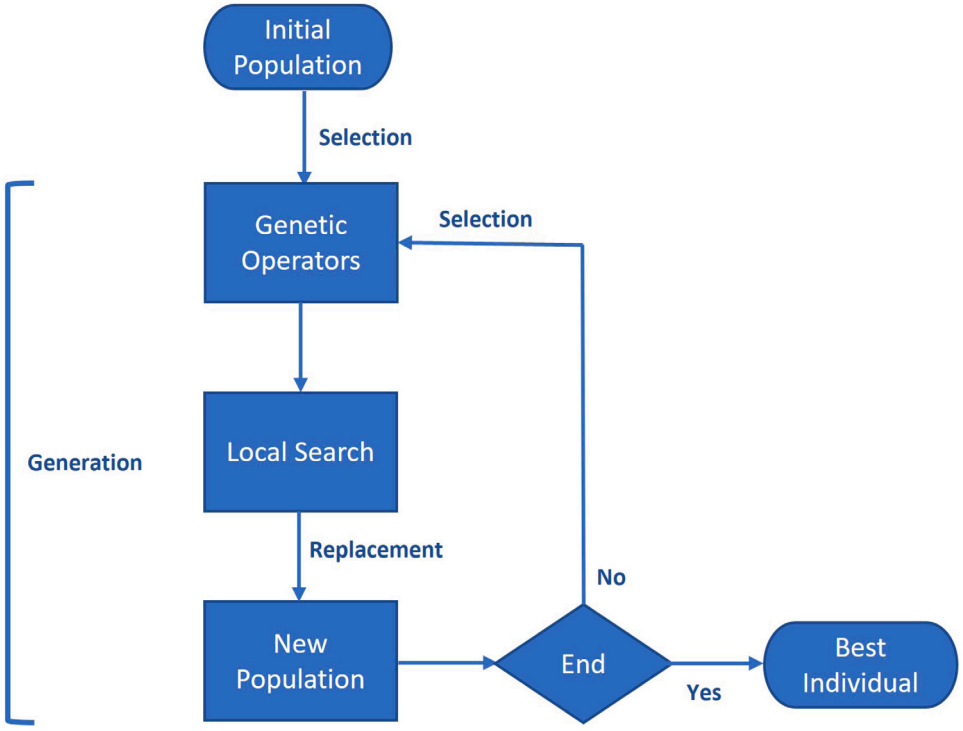

Fig. 1. Typical flowchart of a memetic algorithm.

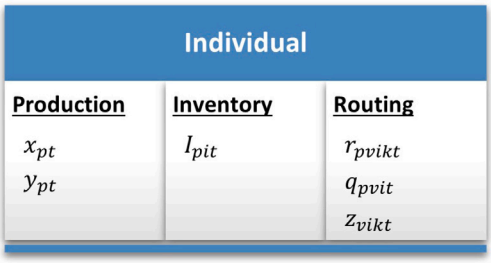

Fig. 2. Structure (class) of a individual.

(iv) **CW**: We also implemented a sequential version algorithm of the (Clarke & Wright, 1964), known as the savings method, which is widely used for the vehicle routing problem (Cordeau et al., 2002), as well as for the PRP (Armentano et al., 2011). This heuristic considers a homogeneous fleet of capacitated vehicles departing from a plant to serve a set of customers. At first, it considers that there is a vehicle serving each customer that will afterward return to the plant. It then determines the savings obtained with the insertion of another customer into the route or even the junction of two routes, provided that the capacity constraints are satisfied. To calculate the savings, the distances between customers are used. The process is continued until all customers are inserted into some route, and no more savings are possible. If the number of vehicles is not enough, a feasibility procedure is again necessary.

(v) **CW-Rand**: To generate the initial population, the parameters were randomly modified for the savings calculation in each period to generate a sufficient and diverse number of solutions. With the modified distance matrix, the CW heuristic is normally applied.

4.4.2. Initial solution: production problem

Next, we present the heuristics implemented to determine the solutions to the production problem.

(i) **WW**: We adapted the Wagner and Whitin (1958) algorithm, which uses dynamic programming to optimally solve the incapacitated lot-sizing problem with a single item over a planning horizon. If the production plan obtained violates the capacity constraints an adjustment procedure must be applied. In this procedure, attempts are made to transfer production to previous periods in order to make the violated constraints feasible.

(ii) **WW-Rand**: To guarantee the generation not only of one but of several individuals for the initial population, we introduced a random component that modifies the preparation costs to ensure the obtained solutions' diversity. From the modified setup costs, the WW heuristic is applied.

(iii) **LpL**: Finally, the lot-for-lot heuristic was implemented to obtain a production plan. This heuristic determines, in each period, the production quantities needed to meet demand without violating capacity constraints. Suppose any lot has not been produced for exceeding the production capacity. In that case, an adjustment procedure will be applied to verify the possibility of anticipating these remaining lots in previous periods, provided that there is still unused capacity.

It is also worth noting that the procedures for adjusting loads and capacity constraints may fail. Thus, we would have the insertion of an infeasible solution in the initial population. We must avoid many infeasible solutions as this could make the search process for the optimal solution to the problem infeasible. On the other hand, some infeasible solutions can help maintain population diversity, avoiding stagnation of the search and allowing for the exploration of the solution space's different regions.

4.5. Selection and replacement mechanisms

From the adopted population structure, we defined two criteria for the selection and replacement of individuals in the population.

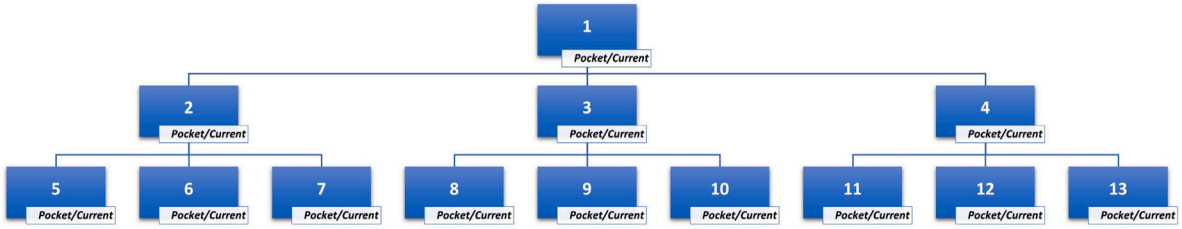

Fig. 3. Population tree structure.

(i) **selectPockets**: In this criterion, only individuals called pockets will be selected. In each sub-population, the lead agent's pocket will be selected for application of the reproduction operators (crossover and mutation) together with the subordinate agents' pocket individuals. Thus, we have four sub-populations, and, in each sub-population, there are three subordinates. Therefore, three recombinations in each sub-population will be performed, generating twelve new individuals at each stage of this process, which we call a new generation;

(ii) **selectCurrents**: In this case, only the current individuals will be selected. As in the selectPockets criterion, the lead agent's current individual will be selected for application of the reproduction operators together with the subordinate agent's current individuals.

At each reproduction operator application, a child individual will be generated that will replace the subordinate agent's current individual. Assuming that agent 1's (leader) pocket has been selected together with the agent 2's (subordinate) pocket, the individual resulting from this operation will replace agent 2's current individual. The replacement criterion will be the same if the selectCurrents criterion is applied. Furthermore, this replacement does not need to be conditioned to the quality of the generated solution as we are assured that the replaced solution will not be better than the selected parents.

4.6. Genetic operators

One of the most significant difficulties in implementing a memetic algorithm is adapting traditional reproduction operators to problems involving a large number of constraints, as in the PRP case. Considering the previously defined selection criteria, we will denote the leader individual by mother and the subordinate individual by father to facilitate the presentation of the operators proposed in this work. It is important to note that, except for the crossOX1 operator, which is a classic TSP operator, the other operators are contributions from this work to the PRP. Among the 5 crossover and mutation operators described below, the crossOPP operator modifies only the production variables, the crossOPR and crossOX1 operators modify only the routing and the others, crossXLF and crossTWP, modify production and routing simultaneously.

(i) **crossXLF**: It is a one-point crossover operator in which the child receives the production plan from the mother and the routing from the father or vice versa. This choice occurs using a drawing with equal probability. The child will be feasible whenever there are no adjustments in the routing inherited from the parents. During the adjustment, there may be delivery anticipations that could affect the original production plan. In any case, if the child is infeasible, a feasibility procedure is applied. According to the defined mutation rate, the mutation operator consists of exchanging two customers in a route chosen at random in the child solution. The operation is detailed in Algorithm 1.

(ii) **crossOPR**: In this operator, described in Algorithm 2, the child receives all the information from the mother and one cut-point, chosen randomly from $[1, T]$, which defines which information from the father's routing will be inherited by the child. The mutation operator will not allow the exchange of routing information for any period within the selected time interval, according to a probability rate. Therefore, the child will remain with the mother's routes for this chosen period.

(iii) **crossOPP**: As described in Algorithm 3, the child will receive all the information from the mother and one cutpoint, selected at random from $[1, T]$, which defines which information from the father's production plan will be inherited. Analogous to crossOPR, the mutation operator will then be applied.

(iv) **crossTWP**: This two-point operator is equivalent to the simultaneous application of the crossOPP and crossOPR operators, i.e., the child will receive the information from the mother, and, once the cut-points are defined, it will receive a part of the production plan (crossOPP) and another part of the routing (crossOPR) from the father. The mutation operator will be applied the same way it is used in the operators individually, i.e., in this case, it will be applied twice.

(v) **crossOX1**: This operator adapts the OX1 (order crossover) proposed by Davis (1985) for the TSP. The child receives all information from the mother, and the OX1 application modifies the routing. Thus, at each period, the vehicle routes are joined to a large route with multiple vehicles. Then, a segment of the mother's route is copied to the child and the rest of the route is completed using the visiting order of the customers present on the father's route. Finally, the routes are distributed among the expanded route's available vehicles, and the vehicle capacities are respected. Note that the number of vehicles used may differ from the parent solutions or even be infeasible.

Algorithm 1: crossXLF

Data: Pocket[i], Pocket[j]
Result: child
if *rand.nextDouble() < 0.5* **then**
  child ← Pocket[i].$(x_{pt}, y_{pt}), \forall p, t$;
  child ← Pocket[j].$(r_{pvikt}, q_{pvit}, z_{vikt}), \forall p, v, i, k, t$;
else
  child ← Pocket[j].$(x_{pt}, y_{pt}), \forall p, t$;
  child ← Pocket[i].$(r_{pvikt}, q_{pvit}, z_{vikt}), \forall p, v, i, k, t$;
end
child.updateInventory();
child.setFitness();
child.setFeasible();
if *not child.isFeasible()* **then**
  child.repair();
end
return child

When a small amount of information is exchanged between individuals, we are, in practice, exploring the neighborhood of these solutions and, therefore, intensifying the search in a given region. On the other hand, when we exchange a lot of information between individuals, there is a process of diversification of solutions, i.e., other regions are reached by the expansion of the search space.

Algorithm 2: crossOPR

Data: Pocket[i], Pocket[j]
Result: child
child ← Pocket[i];
onePoint ← $rand.nextInt() \in [1, T]$;
notExchange ← 0;
for $t \leftarrow$ *onePoint to* T **do**
  if *rand.nextDouble() < taxMutation* **then**
    notExchange = t;
    t ← T+1;
  end
end
for $t \leftarrow$ *onePoint to* T **do**
  if t ! = *notExchange* **then**
    child ← Pocket[j].$(r_{pvikt}, q_{pvit}, z_{vikt}), \forall p, v, i, k$;
  end
end
child.updateInventory();
child.setFitness();
child.setFeasible();
if *not child.isFeasible()* **then**
  child.repair();
end
return child

Algorithm 3: crossOPP

Data: Pocket[i], Pocket[j]
Result: child
child ← Pocket[i];
onePoint ← $rand.nextInt() \in [1, T]$;
notExchange ← 0;
for $t \leftarrow$ *onePoint to* T **do**
  if *rand.nextDouble() < taxMutation* **then**
    notExchange = t;
    t ← T+1;
  end
end
for $t \leftarrow$ *onePoint to* T **do**
  if t ! = *notExchange* **then**
    child ← Pocket[j].$(x_{pt}, y_{pt}), \forall p$;
  end
end
child.updatePlantInventory();
child.repairPlantInventory();
child.setFitness();
child.setFeasible();
if *not child.isFeasible()* **then**
  child.repair();
end
return child

The application of these operators can cause a loss of local or generalized feasibility in individuals, depending on the information that is being changed. Although infeasible solutions are allowed in the population, feasibility procedures need to be applied to minimally preserve population feasibility. Below we describe a summary of such procedures.

4.7. Repair algorithms

The proposed feasibility procedures move the quantities produced, stored, and delivered to earlier or later periods to re-establish the production, inventory, and vehicle load capacities. In this paper, we implement 6 feasibility operators. Among them, we have (i), (ii) and (iii), which make adjustments in the production, while the others promote routing adjustments.

(i) **Production capacity violation**: This procedure performs production transfers to previous periods with unused capacity whenever any violation of the plant's capacity is identified. It should be noted that this procedure may cause a cost increase as periods without production may have to produce small quantities to adjust another period's capacity limit. Furthermore, production anticipation can cause increases related to inventory costs.

(ii) **Negative plant inventories**: This may indicate that the production was delayed, in general, due to the anticipation of delivery after the definition of the initial production plan. In this case, the delayed lot needs to be located and anticipated from the current or previous period. Suppose it is not possible to anticipate the entire lot, avoiding additional preparation costs: in that case, the procedure will try to anticipate the minimum quantity necessary to make the inventory for that period positive.

(iii) **Excess production**: If inventories are not depleted in the last period of the planning horizon, there may have been too much production, and these quantities can be taken directly from the production plan. The crossOPP and crossTWP operators may have caused the excesses. Although these quantities can be used soon and do not directly imply infeasibility, they cause an immediate increase in production costs and inventories.

(iv) **Negative customer inventories**: This may be an indication that the demand was either not met or met late. If the product was delivered late, then this quantity needs to be located and brought forward to a period less than or equal to that identified. Otherwise, this demand needs to be included in another delivery vehicle from the current or previous period. The priority in both cases is the maintenance of existing routes and, if this is not possible, the routes will be modified.

(v) **Excess deliveries**: If the customer's inventories are positive for some product, the procedure, in the last period, makes attempts to remove the excess in some of the deliveries. Excess deliveries may have been caused by operators crossOPR and crossTWP and will cause negative plant inventories.

(vi) **Violation of customer inventory capacity**: This procedure attempts to transfer the excesses to later periods, prioritizing the inclusion in some existing shipments for that customer. Otherwise, the route of a vehicle with unused capacity will be changed.

In the tests carried out, the application of the genetic operators combined with the feasibility procedures, we did not detect a negative impact on the evolutionary process caused by the excess of infeasible individuals in the population of individuals over generations, even though the PRP is highly restricted.

4.8. Local search

For each new feasible individual inserted into the population, we applied a procedure to improve the quality of the solution. This mechanism is associated with the cultural evolution of individuals in memetic algorithms, also called hybridization of the genetic algorithm, as it incorporates specific knowledge of the problem to search for better solutions. The 9 strategies used in this paper are described below. All were applied with the aim of improving the quality of the routes present in the child individual that will be introduced into the population.

(i) **backEmptyVehicle**: This procedure aims to empty the vehicle with the highest unused load capacity from the second half of the planning horizon. The option for the final part of the horizon aims to facilitate possible relocation of this load to previous periods, i.e., anticipating deliveries. Once the vehicle is identified, attempts are made to transfer the load to one or more vehicles in previous periods. If the procedure is successful, the fixed cost for using the vehicle will no longer be computed in

that period. It is important to note that partial emptying of the vehicle may occur since the procedure analyzes the feasibility of transfers per customer.

(ii) **forEmptyVehicle**: Similar to the previous procedure, the objective is to empty the vehicle with the highest unused capacity from the first half of the planning horizon, postponing deliveries. After the vehicle is located, attempts are made to transfer the load to one or more vehicles in later periods.

(iii) **backwardLoad**: A rote was randomly selected from a period in the second half of the planning horizon, and an attempt was made to transfer this route's first customer load backward. If the move is successful and improves the total cost of the solution, a new effort will then be made.

(iv) **forwardLoad**: In this case, attempts are made to postpone the deliveries as long as meeting the demand is not jeopardized. A route from a period in the first half of the planning horizon is drawn, and an attempt is made to transfer this route's first customer load forward. If the move is successful, a new route will then be drawn, and a new transfer analyzed.

(v) **2Opt**: This procedure eliminates two nonadjacent edges and reconnects them in another way. The route must have at least four nodes for its application to be possible. If there is an improvement in the solution, the change is accepted, and the process is repeated. Furthermore, the route and the edges for elimination are randomly chosen.

(vi) **2OptOut**: Unlike the previous version, we now have two edges selected and removed in two different routes, thus dividing the routes into two parts. Then, the initial part of one route is connected to the final part of the other route, thus generating two new routes. Two new routes will be drawn, and the process will be repeated if there is an improvement in the solution.

(vii) **3Opt**: In this procedure, three non-adjacent edges are eliminated and then reconnected differently. Therefore, there are $2^3 - 1$ reconnecting possibilities, and the route must have at least six nodes. The route and edges are randomly chosen, and the process continues as long as the solution improves.

(viii) **swapCustomer**: In this case, in a randomly selected route, two customers on said route are drawn and swapped.

(ix) **relocateCustomer**: A randomly selected customer is removed from a route and reinserted into the same route, but in a different position from the original one, or even inserted into another route from the same or different periods.

Although they are classic heuristics in the context of routing problems, these strategies or their variations have been implemented for the PRP in different situations. We highlight the local searches proposed by Senoussi et al. (2018), which use more sophisticated versions of (iii) and (iv). In addition, we have some of the other strategies implemented by Yagmur and Kesen (2021) for the PRP considering production scheduling and heterogeneous fleets, and also the work of Manousakis et al. (2022), which considers one plant and one product.

4.9. Restart and diversification strategies

Whenever the evolutionary process converges to an optimal location, stagnating the search for new solutions, we need strategies to explore the search space's other regions. To monitor the convergence of individuals over generations, we defined two criteria to determine population diversity. The first, discrete, determines the number of individuals with different fitness values. The second, continuous, determines the distance between the current population's smallest and the largest fitness values.

While adjusting the parameters, we observed that only resetting the population through the strategies used to generate the initial population was not efficient in guaranteeing a new population capable of allowing for the exploration of different regions. We then used the same procedures from the local search but now adapted to change the individuals with repeated and combined movements between the different adopted strategies. This promotes the necessary diversity among the new population's individuals and restarts the search from other points in the search space. This way, whenever the diversity criteria reach critical levels, the population can then be restarted through these strategies.

With the definition of the basic components, an overview of the proposed memetic algorithm is presented below.

4.10. Basic structure

Initially, the twenty-six individuals that will compose the initial population are generated (see Fig. 2),using the procedures described in Section 4.4. If a solution is infeasible, a procedure described in Section 4.7 to make this solution feasible is then applied. If the procedure fails, the individual will be inserted into the population even if it is infeasible. Then, the evaluation function (Section 4.2) is applied to each of the new individuals and the initial population is ordered according to the population structure (Section 4.3). Diversity rates are calculated and from the established selection mechanisms (Section 4.5), crossover and mutation operators (Section 4.6) are applied. The newly generated individuals that are infeasible go through the feasibility process again (Section 4.7). After, the local search (Section 4.8) is applied only on the feasible descendants. This way, the population is reorganized and the diversity rates (Section 4.9) are updated. If the diversity indices reach the established critical level then the population will be restarted (Section 4.9) and restructured according to the fitness values. Finally, if the stopping criterion is reached, the algorithm will return the best solution found. This solution will be stored in agent 1's pocket individual. Otherwise, new individuals will be selected for operator application and the evolutionary process will continue to be repeated, as shown in Fig. 1 and in Algorithm 4.

Algorithm 4: Basic structure of MA

```
Data: Parameters - Instance and MA
Result: Best individual at population
Initialize population;
Update population structure;
Evaluate diversity;
while stop == false do
    forall Population do
        if rand.nextDouble() < 0.9 then
            A,B ← selectPockets(population);
        else
            A,B ← selectCurrents(population);
        end
        child ← GeneticOperators(A, B);
        if child is not feasible then
            Repair(child);
        end
        if child is feasible then
            LocalSearch(child);
        end
    end
    Update population structure;
    Evaluate diversity;
    if critical diversity achieved then
        Restart population;
        Update population structure;
        Evaluate diversity;
    end
    if stop criterion reached then
        stop ← true;
    end
end
return Best individual;
```

Table 2
Parameters for generation of the multiple item instances.

Customers demand	$d_{pit} \in [10, 100]$
Inventory level upper bound at the customers	$U_{pi} = \lfloor (\sum_t d_{pit}/T) \cdot \lambda \rfloor$, where $\lambda \in \{2, 3, 4, 5\}$
Initial inventory at the plant	$I_{p00} = 0$
Initial inventory at the customers	$I_{pi0} = \sum_{t=1}^{\tau} d_{pit}$, where $\tau \in \{1, 2, 3, 4\}$
Unit inventory cost at the plant	$h_{p0} = 3$ and 8
Unit inventory cost at the customers	$h_{pi} \in [1, 10]$
Production capacity	$B = \lfloor 3 \cdot (\sum_p \sum_i \sum_t d_{pit}/T) \rfloor$
Unit production time	$b_p = 1$
Production cost	$c_p = 10 \cdot h_{p0}$
Setup production cost	$s_p = 100 \cdot h_{p0}$
Coordinates	$x_k \in [0, 500] \quad y_k \in [0, 1000]$
Vehicle capacity	$C = P \cdot (\max_{\forall p,i} U_{pi})$
Transportation cost	$a_{ik} = \lfloor \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2} + 0.5 \rfloor$
Fixed transportation cost	$f = (N + 1) \cdot (\max_{\forall i,k} a_{ik})$

It is important to note that the approach could be adapted for the multi-product PRP setting with a heterogeneous fleet. Following is presented an analysis of the results obtained by the MA in a set of randomly generated instances for the multi-product PRP. Furthermore, we compared its performance with the state-of-the-art approach involving a set of classic PRP instances with a single item.

5. Computational experiments

The proposed MA and its variants were implemented in the Java language, JDK version 14.0.2, using the commercial solver Gurobi, version 9.1.2, and also the Java interface available for the environment. The tests were performed on a computer with an Intel Core i 7-10510U processor (2.3 GHz, 8 MB cache, quad-core), 16 Gb RAM DDR4 2666MHz, with Windows 10 Home operating system.

As far as we know, it was not possible to find a set of instances with multiple products that could be used as a benchmark for this research. Therefore, we randomly generated a set of instances with multiple products based on the instances by Archetti et al. (2011) and with parameters similar to those used in previous research, such as that of Armentano et al. (2011) and Mostafa and Eltawil (2019). It is important to point out that we make available, through the supplementary material, all the instances used in this article, in addition to the results obtained, to facilitate future analyses and comparisons.

In the following subsection, we describe the generation of instances with multiple products in detail. In Section 5.2 we present a summary of the tests performed to obtain the proposed MA's best parameter combination. In Section 5.3 we show, through some experiments, the effect of the local search on the algorithm developed in this paper. In Section 5.4 we have the results obtained for instances with multiple items. Finally, in 5.5 we compare the results for a classic set of PRP instances with a single item.

5.1. Instance generation

Table 2 details the parameters used to generate instances with multiple products. Regarding the literature, the main changes proposed are found in the parameters used to generate inventory capacities, including initial customer inventory capacities and vehicle capacity. For the other parameters, we used, as references, research carried out by Archetti et al. (2011) and Armentano et al. (2011).

Table 3 presents a general summary of the four groups of instances with 5, 10, 20 and 30 customers and 3, 5, 8 and 10 products, respectively. The first two groups have 6 periods and the other two have 12 periods. Furthermore, the first group of instances only has 1 vehicle, the second 2 and the following two 5 vehicles. The demands are variable, and the initial customer inventories are not zero. The plant's production capacity is limited, and the storage capacity is unlimited, but the initial inventories are zero.

Table 3
Overview of the multiple item instances.

Instances	5C	10C	20C	30C
Number of instances	200	200	200	200
Number of periods	6	6	12	12
Number of products	3	5	8	10
Number of customers	5	10	20	30
Number of vehicles	1	2	5	5
Number of seeds	5	5	5	5
Demand	Variable	Variable	Variable	Variable
Production capacity	Constant	Constant	Constant	Constant
Plant inventory capacity	Unlimited	Unlimited	Unlimited	Unlimited
Customers inventory capacity	Constant	Constant	Constant	Constant
Initial inventory at the plant	0	0	0	0
Initial inventory at the customers	Variable	Variable	Variable	Variable
Vehicle capacity	Constant	Constant	Constant	Constant

Table 4
Descriptions of the multiple item instance classes.

Class	Type	Descriptions
Class I	1–10	Standard instances
Class II	11–20	High production costs (Class I × 10)
Class III	21–30	Large transportation costs (Class I × 5)
Class IV	31–40	No customer inventory costs

Like Archetti et al. (2011), we split the groups into four classes according to Table 4. Class I (instances of 1 to 10) has the basic configuration of production, inventory and transport costs, serving as a basis for the generation of the others. Class II (11 to 20) has high production costs, which are equivalent to the costs of Class I multiplied by 10, Class III (21 to 30) has high transport costs, i.e., the costs will be 5 times higher than Class I. Finally, Class IV (31 to 40) has no customer inventory costs. Each class has 10 instances with 5 seeds each; therefore, we have 200 instances for each of the groups, totaling 800 new instances in the set as a whole.

5.2. Best parameters values and strategies

We performed several preliminary tests using samples of instance subsets proposed herein to obtain the best combination of parameters and strategies for maximum MA performance.

In the initial population generation, we evaluated the diversity and quality of the solutions obtained by the techniques separately or by their combinations. We chose the sequences with the heuristics described in Algorithm 5 to generate the population of 26 individuals.

Algorithm 5: Initialize population

```
Data: Instance parameters
Result: Population
pocket[1] ← CW() + WW();
pocket[2] ← BFD-Inverse() + WW();
current[1] ← WW() + BFD();
current[2] ← CW() + LbL();
for i ← 3 to 13 do
    pocket[i] ← WW-Rand() + CW-Rand();
    current[i] ← BFD-Rand() + WW-Rand();
end
return Population
```

The individual selection mechanisms were applied according to a probability rate; thus, the criterion that selects only pockets (selectPockets) was applied with a rate of 90%; if not, the second criterion (selectCurrents), which only selects the currents, will be applied according to Algorithm 4.

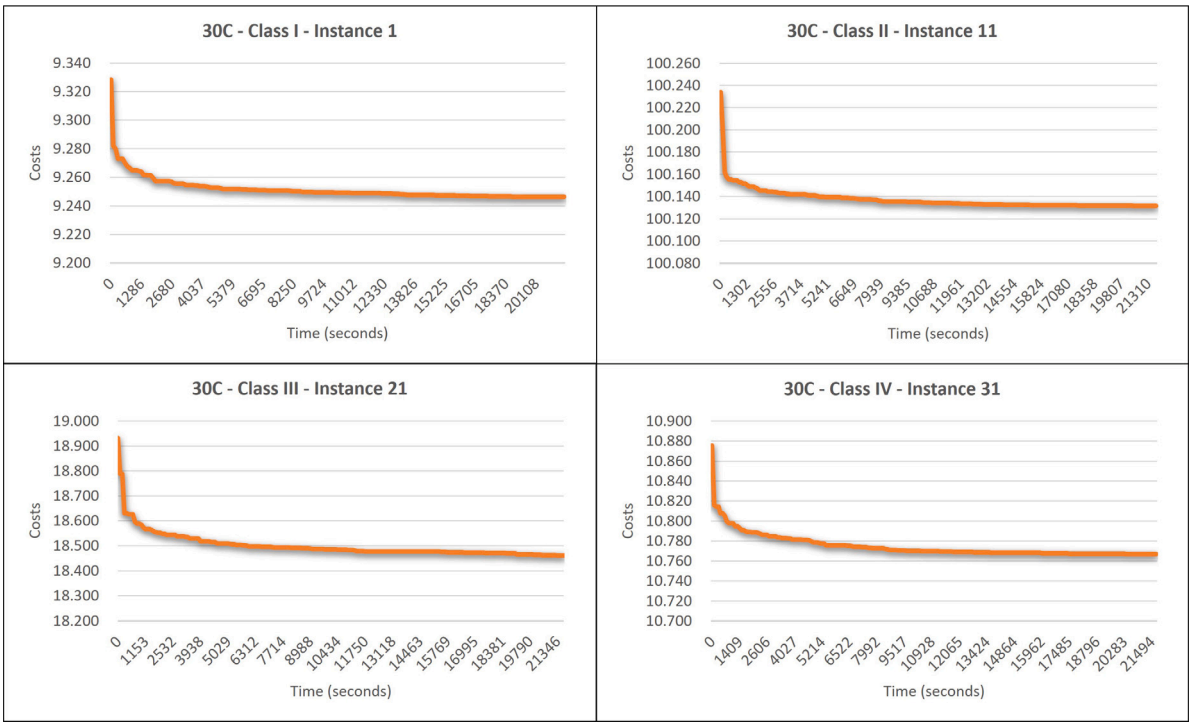

Fig. 4. Behavior of MA for the instances with 30 customers (stopping criterion was 21,600 s).

The reproduction operators also had their efficiency evaluated individually or combined within the evolutionary process. The best results were achieved by applying all operators equally, one at a time over generations. The mutation was used with a 10% probability rate on all operators.

Several procedure combinations with different intensification levels were tested regarding the local search. The best performance was achieved using all strategies with 10 intensification iterations, i.e., all strategies were applied sequentially, one after the other, and this process was repeated 10 times.

Finally, when the diversity indices reach critical levels, specifically when the continuous diversity rate is less than 100 or when the discrete rate is less than 4, the population diversification mechanism is activated. Furthermore, after 100 generations without updating the incumbent, a new initial population is generated, restarting the search for a new population.

In Fig. 4, we can observe the behavior of the MA through the evolution of the incumbent (fitness versus time) over 6 h of execution in a sample with 4 instances, the first from each class of group 30C. Although the most significant cost reduction occurs within the first hour of run time (3600 s), we can see that the MA continues to evolve, albeit more slowly, over the 6 h of execution. This behavior can also be verified in group 20C; however, the cost reduction was more pronounced in the first hour of run time. From these analyses, we can infer that the ideal stopping criterion for larger instances would be 2 and 3 h for groups 20C and 30C. The evolution of the incumbent in groups with 5 and 10 customers stagnated within the first hour of execution. Given this scenario and seeking a balance between the number of instances and the computational resources available to execute the tests, we defined the maximum time of 3600 s (1 h) as a stopping criterion for the solver and the MA in all instances groups.

5.3. Genetic versus memetic algorithm

Some experiments were conducted to analyze the local search's impact on the evolutionary process. Without the incorporation of these strategies, we would have a genetic algorithm. Thus, in Table 5 and Fig. 5 we compare the genetic and memetic algorithms for all instances with 5 customers, limiting the maximum execution time to 5 min. These results show the impact of the local search on the proposed method since the average gap obtained by the GA is practically double that of the MA. The average gap, in percentage, represents the difference between the objective function value obtained by the method and the lower bound obtained by the solver, divided by the lower bound. Thus, when analyzing the performance in each class, we have that the closest result between the techniques occurs in class IV, but with the GA achieving results that were, on average, 40% worse.

Table 5
The MA versus GA average gap in percentage for the instances with 5 customers (stopping criterion was 300 s).

	Classes				Average
	I	II	III	IV	
GA	1,83	0,20	2,46	1,87	**1,59**
MA	0,50	0,07	1,12	1,34	**0,76**

In Fig. 5, we can observe the average behavior of the 5 seeds per instance. It is, therefore, possible to notice that the closest performance between the techniques occurs in instances from Class II (11 to 20) and in instance 35. In the other instances, however, the performance difference between the techniques is quite pronounced. Class II has instances with high production costs, which strongly reduces the impact of the improvements obtained in routing through local search, i.e., the closest behavior between the methods in these instances was expected.

To graphically illustrate the effect caused by the local search, we randomly chose an individual (child) from the population right after the application of the reproduction operator in the first generation. For this, we selected instance 1 from the set with 14 customers proposed by Archetti et al. (2011) to facilitate the visualization of routes involving only one vehicle and one product. After 10 iterations, the cost of the initial solution, which was 46,907, was reduced to 43,369. This improvement can be easily seen in Figs. 6 and 7, more specifically in the intersection-free routes of periods 2 (b), 3 (c), 4 (d) and 5 (e) that were obtained after applying the local search.

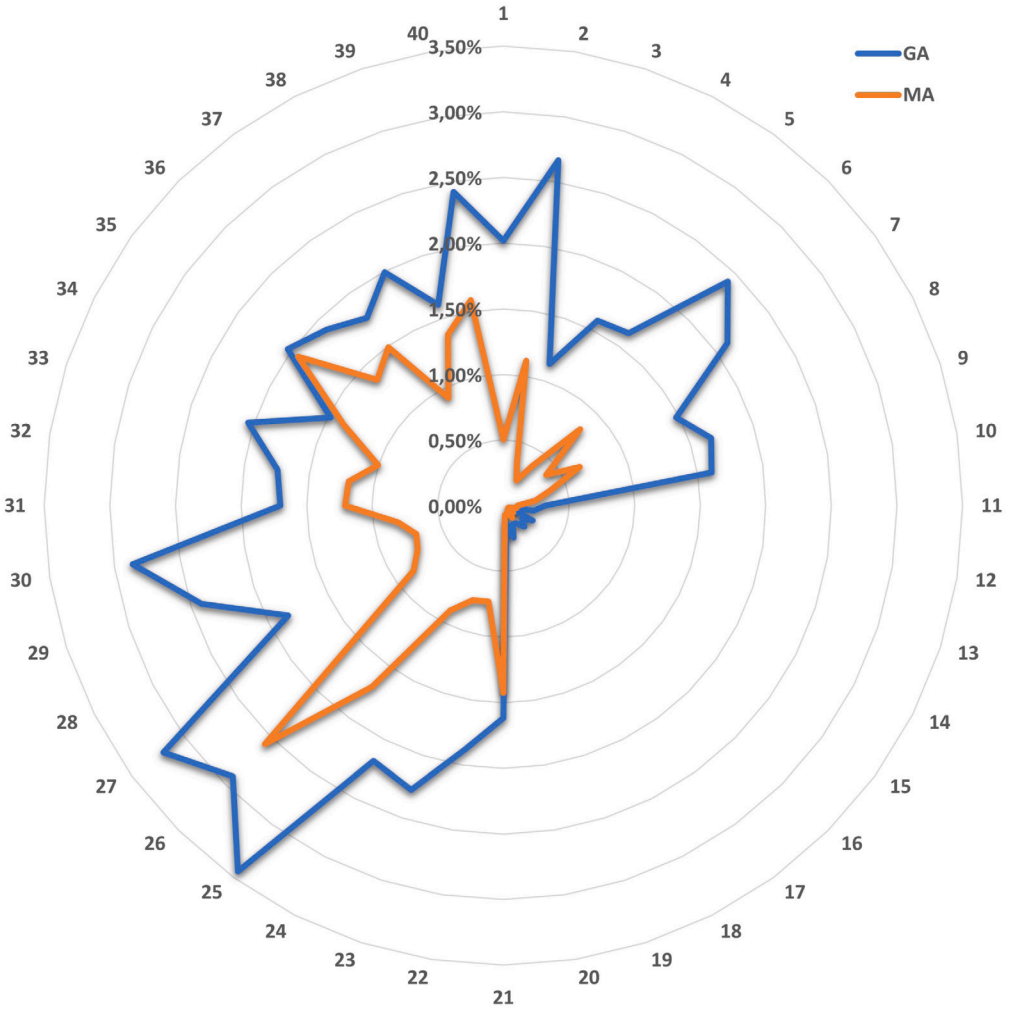

Fig. 5. The MA versus GA average gap in percentage for the instances with 5 customers (stopping criterion was 300 s).

5.4. Computational results for the multiple item instances

Table 6 presents a general summary of the MA performance compared to the values obtained by the Branch-and-Cut using the Gurobi solver at the time limit of 3600 s. In this table, for each instance class, the average gap in percentages represents the difference between the objective function value obtained by the method and the lower bound obtained by the solver, divided by the lower bound.

For the instance group with 5 customers (5C), the two methods obtained excellent results in all instances. In Classes I and III, both found the optimal solutions in all instances, while in Class II the average gap was equal to zero both in the solver and in the MA. However, the average gap in the two seeds was equivalent to 0.01% for both. In Class IV, the two methods presented the worst performance, but the average values were not higher than 0.05%. As shown in Table 8, the solver did not determine the optimal solution in only 3 instances from Class IV. In comparison, the MA found the optimal solution in 188 instances, equivalent to 94% of the total for this group. Regarding the computational times shown in Table 7, the solver solved, on average, all instances in 224 s. More specifically, the first two classes were solved in less than 15 s, while Class II took around 2 min, and in Class IV, the average times were about 12 min. It should be noted that the MA exhausted the computational time limit defined as the stopping criterion in all instances for this group.

In the group with 10 customers (10C) we can see that the MA obtained very similar results to the solver in all classes, the difference between the two techniques being only 0.01% in the overall average. Unlike the previous group, the class that presented the worst average gap was Class III with an average of around 3%. The best performance was obtained in Class II, with high production costs and an average gap of 0.10%. Classes I and IV also presented very good results, with average gaps of less than 1%. So, in general, the average results were close to the optimal solutions in this group. Regarding computational times, we emphasize that the techniques were also very close as Gurobi determined only one optimal Class I solution. In the others, however, it used all the available processing time.

As seen in Table 6, the MA performance was notably superior for groups of large instances, with 20 and 30 customers. At the same time, Gurobi did not find any feasible solution within the established time limit. On the other hand, the MA obtained feasible solutions with excellent average gaps, especially in Class II, which presented average values lower than 0.35%. In group 20C, Classes I and IV had results close to 3% and 3.5%, respectively, while Class III was around 5%. In general, and concerning the classes, the MA performance was similar to the two groups of smaller instances. The results from group 30C were even better, showing the proposed method's robustness. The values obtained for Class II indicate that the solutions determined by the MA were very close to the optimal solutions, with gaps less than 0.25%. In Classes I and IV, the gaps were around 2% and 2.7%, respectively. In Class III the average gap was 5.8% while the general average was 2.7%. No optimal solution was obtained within the set time limit.

Faced with the solver's difficulty in obtaining feasible solutions for instances with 20 and 30 clients, we increased the maximum execution

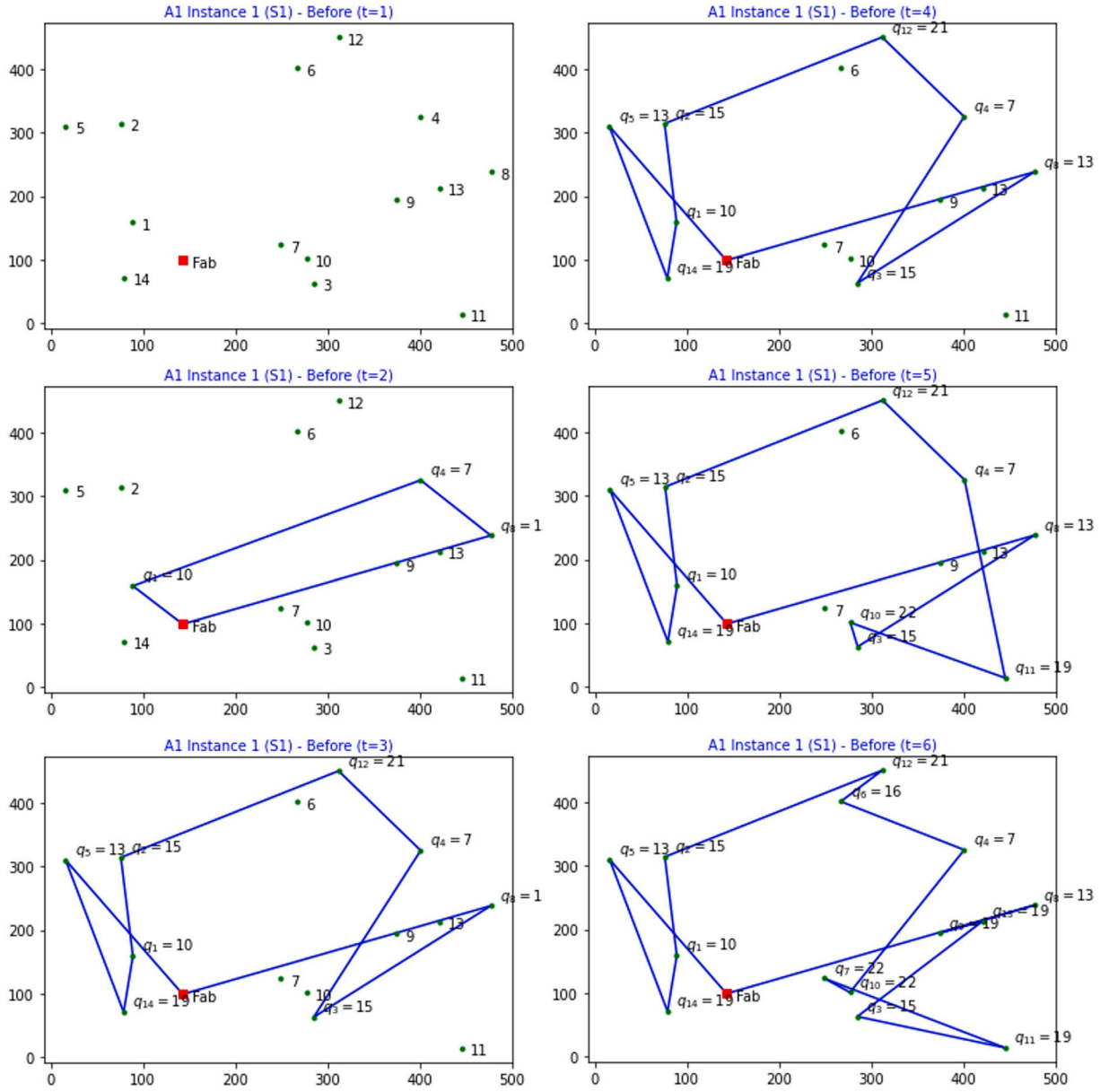

Fig. 6. Routes **before** the application of the Local Search on the single item instance from Archetti et al. (2011).

time to two hours (7200 s) and changed the solver's default configuration to focus on finding feasible solutions. Average run times are shown in Table 7. Furthermore, in Table 9 we can observe that there was a significant increase in the number of feasible solutions obtained by Gurobi, except in Class IV. On the other hand, we can observe in Table 10 that the quality of these solutions are not competitive when compared to the MA.

5.4.1. Hybrid strategy using solver

Faced with the solver's difficulty in obtaining feasible solutions for instances with 20 and 30 customers, we decided to explore the solver's ability to improve an initial solution obtained by the MA in greater depth. This strategy is commonly called the hybridization process, which combines a metaheuristic with the exact method used by the solver.

In Table 10, we present the results of these experiments using the same stopping criterion as the previous tests, i.e., the total execution time was 1 h. Based on this criterion, the MA was run for 5 min and the solver for 55 min. In other words, at the end of the first 5 min, the best solution obtained by the MA was provided as the initial solution for the solver, which was run for another 55 min.

As can be seen, except for the instance group of 10 customers in which the three strategies achieved practically the same results, the hybrid strategy achieved better solutions than the MA and the solver alone in the other groups with 20 and 30 customers. On average, it improved the solutions of the group with 20 customers by 0.74 percentage points and the solutions of the group with 30 customers by 0.26 percentage points. Analyzing the results by classes, we observed that in the group with 20 customers, the hybridization strategy managed to improve the results for all classes, while in the group with 30 customers, this improvement occurred only in classes III and IV.

To evaluate the performance of the techniques, we used the tool proposed by Dolan and More (2002) to facilitate the visualization and interpretation of the computational results. In Fig. 8, we compare Gurobi and MA based on the obtained gaps. Thus, P is the set with 705 instances where both techniques found feasible solutions to the problem, S is the set with both techniques and τ is the metric used.

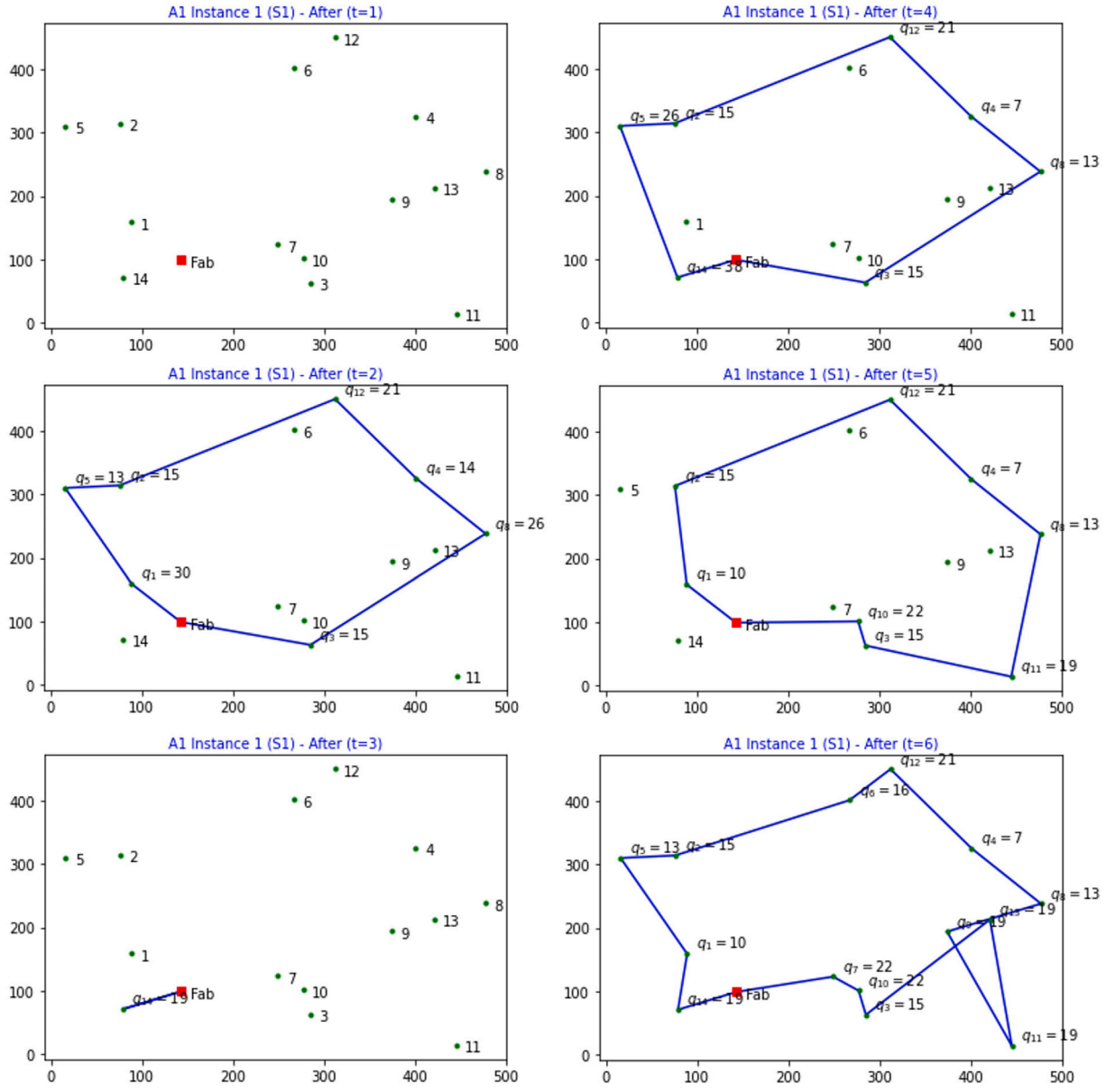

Fig. 7. Routes **after** the application of the Local Search on the single item instance from Archetti et al. (2011).

Therefore, the solution quality is calculated by

$$r_{ps} = \frac{t_{ps}}{\min\{t_{ps} : s \in S\}}$$

that is, the obtained gap divided by the minimum value found by the techniques. In addition, t_{ps} is the gap obtained by the technique in S to solve instance p. Considering N_p as the total number of instances, the probability $P_s(\tau)$ of each technique being better or equal to τ is given by

$$P_s(\tau) = \frac{|p \in P : r_{ps} \leq \tau|}{N_p}$$

In $\tau = 1.00$, we have the percentage of instances in which technique S obtained the best solution. Thus, we can observe that, concerning the gaps, MA found the best solution in approximately 83% of the instances compared to Gurobi, i.e., obtained the best solution in 585 out of a total of 705 instances.

Fig. 9 presents a comparison between the initial solution used by the solver and that obtained by the MA after 5 min of execution, in addition to the final result after 55 min of execution using Gurobi. From the box plots of each instance group, we can observe that the larger the instances, the greater the difficulties encountered by the solver to improve the solutions obtained by the MA. Even so, this hybridization strategy showed advantageous in getting better results than those achieved by MA or Gurobi alone. These results also show that the combination of exact methods and the metaheuristic proposed in this paper could be a promising path.

5.5. Computational results for the single item instances

Although the development of the proposed MA in this article focuses on the multi-product PRP, we chose to compare the performance of the MA with other techniques using PRP instances with a single item, given that they are widely used in the literature.

Given a large number of instances in the set proposed by Archetti et al. (2011) and to enable comparisons with other papers published more recently, we selected these sets for comparison in this study, as shown in Table A.14.

The gap(%) for each instance is calculated as the difference between the solution obtained by the method and the best solution known so

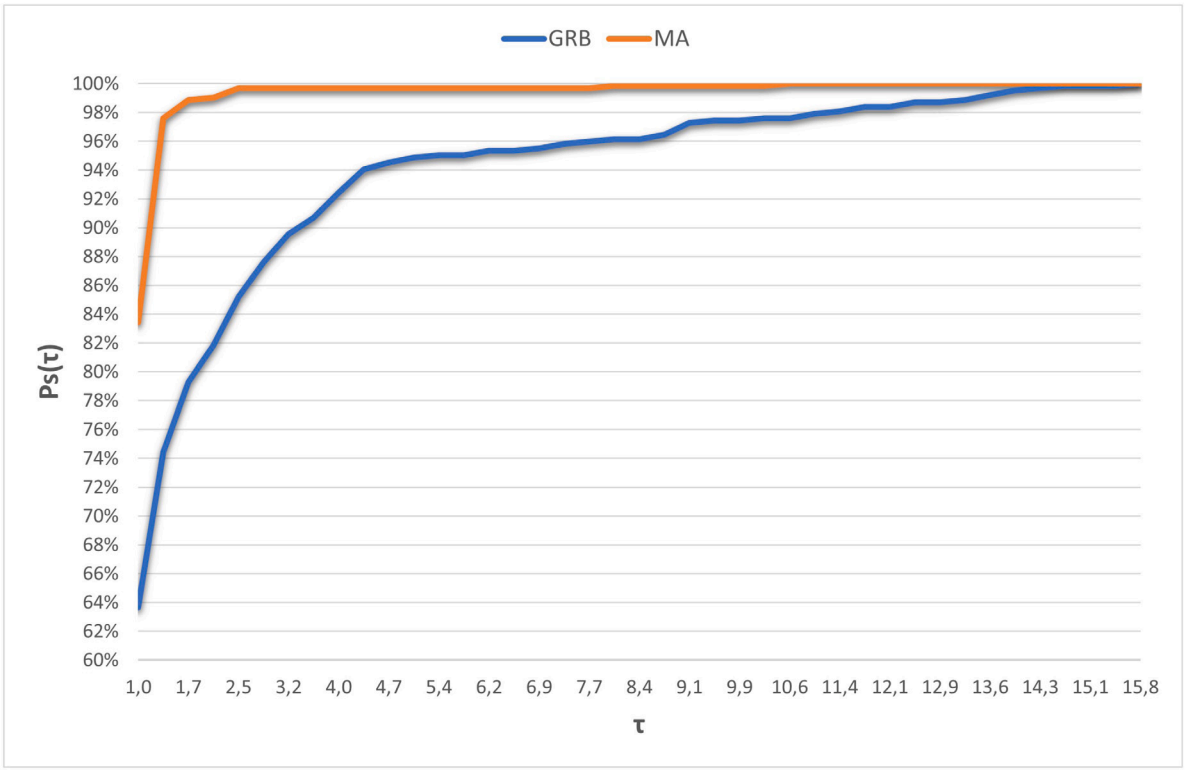

Fig. 8. Comparative analysis of the performance of techniques using the gaps obtained.

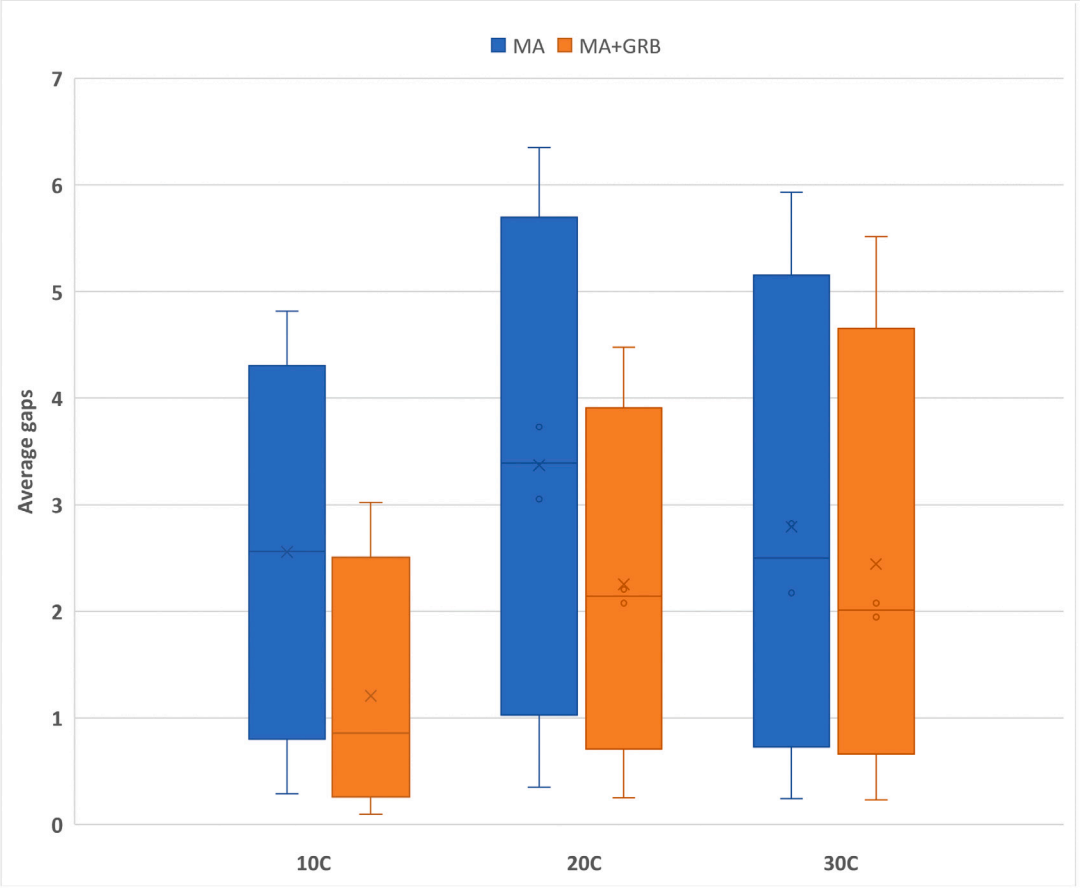

Fig. 9. Box-plots comparing the improvement achieved by the hybridization strategy (average gap in percentage).

Table 6
The average gap in percentage for the multiple item instances.

Set	Method	Classes	Seeds					Average
			1	2	3	4	5	
5C	Gurobi	I	0,00	0,00	0,00	0,00	0,00	**0,00**
		II	0,00	0,00	0,00	0,01	0,01	**0,00**
		III	0,00	0,00	0,00	0,00	0,00	**0,00**
		IV	0,00	0,05	0,03	0,03	0,00	**0,02**
		Average	**0,00**	**0,01**	**0,01**	**0,01**	**0,00**	**0,01**
	MA	I	0,00	0,00	0,00	0,00	0,00	**0,00**
		II	0,00	0,00	0,00	0,01	0,01	**0,00**
		III	0,00	0,00	0,00	0,00	0,00	**0,00**
		IV	0,00	0,05	0,05	0,05	0,01	**0,03**
		Average	**0,00**	**0,01**	**0,01**	**0,01**	**0,00**	**0,01**
10C	Gurobi	I	0,72	0,77	0,89	0,59	0,78	**0,75**
		II	0,09	0,10	0,09	0,09	0,10	**0,09**
		III	2,83	3,25	2,87	2,55	3,55	**3,01**
		IV	0,94	1,04	0,76	0,87	1,19	**0,96**
		Average	**1,15**	**1,29**	**1,15**	**1,03**	**1,40**	**1,20**
	MA	I	0,72	0,77	0,89	0,59	0,78	**0,75**
		II	0,09	0,11	0,09	0,10	0,10	**0,10**
		III	2,84	3,31	2,89	2,57	3,54	**3,03**
		IV	0,97	1,07	0,80	0,89	1,15	**0,98**
		Average	**1,15**	**1,31**	**1,17**	**1,04**	**1,39**	**1,21**
20C	Gurobi[a]							
	MA	I	3,05	2,73	2,69	2,47	2,87	**2,76**
		II	0,32	0,37	0,31	0,34	0,33	**0,33**
		III	5,33	5,48	4,91	6,02	5,33	**5,42**
		IV	3,26	3,29	3,57	3,74	3,25	**3,42**
		Average	**2,99**	**2,97**	**2,87**	**3,14**	**2,95**	**2,98**
30C	Gurobi[a]							
	MA	I	1,92	2,05	2,16	1,80	2,33	**2,05**
		II	0,22	0,27	0,23	0,22	0,24	**0,23**
		III	5,88	5,79	6,02	6,03	5,30	**5,80**
		IV	2,69	2,56	2,54	2,67	2,63	**2,62**
		Average	**2,68**	**2,67**	**2,74**	**2,68**	**2,62**	**2,68**

[a]Any feasible solution was found after 1 h of run time.

Table 7
The average computing time in seconds for the multiple item instances.

Set	Method	Classes				Average
		I	II	III	IV	
5C	Gurobi	4	13	122	756	**224**
	MA	3.600	3.600	3.600	3.600	**3.600**
10C	Gurobi	3.555	3.600	3.600	3.600	**3.589**
	MA	3.600	3.600	3.600	3.600	**3.600**
20C	Gurobi	3.600	3.600	3.600	7.200	**4.500**
	MA	3.600	3.600	3.600	3.600	**3.600**
30C	Gurobi	4.321	4.912	5.470	7.200	**5.476**
	MA	3.600	3.600	3.600	3.600	**3.600**

Table 8
The number of optimal solutions for the multiple item instances.

Set	Method	Classes				Total
		I	II	III	IV	
5C	Gurobi	50	50	50	47	**197**
	MA	50	48	49	41	**188**
10C	Gurobi	1	–	–	–	**1**
	MA	–	–	–	–	–
20C	Gurobi	–	–	–	–	–
	MA	–	–	–	–	–
30C	Gurobi	–	–	–	–	–
	MA	–	–	–	–	–

Table 9
The number of feasible solutions for the multiple item instances.

Set	Method	Classes				Total
		I	II	III	IV	
5C	Gurobi	50	50	50	50	**200**
	MA	50	50	50	50	**200**
10C	Gurobi	50	50	50	50	**200**
	MA	50	50	50	50	**200**
20C	Gurobi	50	50	50	5	**155**
	MA	50	50	50	50	**200**
30C	Gurobi	50	50	50	–	**150**
	MA	50	50	50	50	**200**

Table 10
The average gap in percentage for the hybridization's strategy.

Set	Method	Classes				Average
		I	II	III	IV	
10C	Gurobi	0,75	0,09	3,01	0,96	**1,20**
	MA-Hybrid	0,75	0,10	3,02	0,97	**1,21**
	MA	0,75	0,10	3,03	0,98	**1,21**
20C	Gurobi	3,57	0,45	10,64	11,26	**6,48**
	MA-Hybrid	2,21	0,25	4,48	2,08	**2,25**
	MA	2,76	0,33	5,42	3,42	**2,98**
30C	Gurobi	13,44	2,34	20,94	–	**12,24**
	MA-Hybrid	2,08	0,23	5,52	1,95	**2,44**
	MA	2,05	0,23	5,80	2,62	**2,68**

far, divided by the best solution. To determine the best-known solution in each instance, including the solutions obtained in this work, we used the supplementary material made available by the authors whose works are listed in 11. We used the most relevant research from the last decade, emphasizing recent research by Manousakis et al. (2022), which presented excellent results for this set of instances. Another important highlight of these research articles is that they all use exact methods combined with heuristics.

We can see in Table 12 that the MA achieved excellent results for the group with 14 customers, compared to the other methods, given that the method proposed in this work presented better results than the ACJ-ALNS, QWXFP-VNS and CCJ-M methods. On average, for this set of instances, the best results were achieved by MKKZ-M.

In Table 13, the computational times of all methods are displayed. It is worth noting that Vadseth et al. (2022) calculated an estimate for the MKKZ-M method times since the authors did not disclose the total execution time of the metaheuristic in the original research. We used 300 s as a stopping criterion in MA for group A1. In the MA-Hybrid, we use 1/5 of the running time for the MA and 4/5 for the Gurobi solver. For groups A2 and A3, we used 3.600 s, with the division of the time the same used in the A1 group. As we can see, the QWXFP-VNS method presented the best average computational times, followed by the SS-M method. Below we present conclusions and some future perspectives.

6. Conclusions

This article presents a new Memetic Algorithm with novel operators to deal with the Multi-Product Production Routing Problem. Based on extensive computational experiments, it was possible to conclude that the strategies proposed outperform a state-of-the-art solver in determining feasible high-quality solutions, mainly in large instances of the problem. The proposed Memetic Algorithm achieved excellent results compared to the solver's in sets with 5 and 10 customers. More specifically, it optimally solved almost all instances with 5 customers. Furthermore, the comparison with the lower bounds obtained by the

Table 11
An overview of state-of-the-art PRP approaches for the single item instances.

Reference	Abbreviation	Approach	Solver	Data set
Adulyasak et al. (2014)	ACJ-ALNS	MIP-based ALNS	CPLEX 12.2	A
Solyalı and Süral (2017)	SS-M	MIP-based heuristic	CPLEX 12.5	A
Qiu et al. (2018a)	QWXFP-VNS	MIP-based VNS	CPLEX 12.6	A
Chitsaz et al. (2019)	CCJ-M	Decomposition Matheuristic	CPLEX 12.6	A
Li et al. (2019)	LCCZ-M	MIP-based heuristic	CPLEX 12.6	A
Avci and Yildiz (2019)	AY-M	MIP-based Local Search	CPLEX 12.6	A
Manousakis et al. (2022)	MKKZ-M	MIP-based Local Search	Gurobi 9.0.2	A
This paper	MA	Memetic Algorithm	Gurobi 9.1.2	A
This paper	MA-Hybrid	MA + Gurobi	Gurobi 9.1.2	A

Table 12
The average gap in percentage for best known solutions for data set A.

Set	Class	ACJ-ALNS	AADF-IM	SS-M	QWXFP-VNS	CCJ-M	LCCZ-M	AY-M	MKKZ-M	VACS-M	MA	MA-Hybrid
A1	I	1,70	–	0,03	0,28	0,47	0,15	**0,01**	0,05	–	0,41	0,03
	II	0,37	–	**0,01**	0,05	0,08	0,03	**0,01**	**0,01**	–	0,06	0,02
	III	8,42	–	0,18	1,52	2,20	0,75	**0,04**	0,38	–	1,50	0,05
	IV	0,93	–	0,03	0,56	0,23	0,08	0,01	**0,00**	–	0,28	0,02
	Average	2,85	–	0,06	0,60	0,75	0,25	**0,02**	0,11	–	0,56	0,03
A2	I	1,37	0,37	0,29	0,28	0,32	0,23	0,20	**0,04**	0,06	2,80	2,21
	II	0,22	0,08	0,06	0,08	0,08	0,06	0,06	**0,00**	0,02	0,26	0,23
	III	4,47	1,64	1,00	0,95	1,24	0,90	0,86	**0,14**	0,23	4,22	3,39
	IV	0,39	0,25	0,26	0,18	0,19	0,22	0,21	0,10	**0,03**	2,30	1,50
	Average	1,61	0,59	0,40	0,37	0,46	0,35	0,33	**0,07**	0,09	2,40	1,83
A3	I	1,14	0,30	0,21	0,31	0,40	0,16	0,16	0,28	**0,04**	3,11	2,45
	II	0,34	**0,04**	0,22	0,26	0,22	0,21	0,22	0,17	0,17	0,61	0,53
	III	4,29	1,62	0,77	0,93	2,22	0,78	0,86	0,67	**0,12**	5,84	4,73
	IV	0,51	0,25	0,24	0,27	0,25	0,23	0,27	0,31	**0,08**	2,39	1,54
	Average	1,57	0,55	0,36	0,44	0,77	0,34	0,38	0,36	**0,10**	2,99	2,31

Table 13
The average computing time in seconds for the data set A.

Set	Class	ACJ-ALNS	AADF-IM	SS-M	QWXFP-VNS	CCJ-M	LCCZ-M	AY-M	MKKZ-M[a]	VACS-M	MA	MA-Hybrid
A1	I	9	–	5	12	18	29	12	86	–	300	115
	II	9	–	5	13	18	28	10	86	–	300	93
	III	9	–	5	12	17	28	9	105	–	300	139
	IV	9	–	5	13	18	25	9	98	–	300	115
	Average	9	–	5	12	18	28	10	94	–	300	116
A2	I	50	339	16	25	400	43	89	1513	130	3600	3600
	II	50	236	14	21	344	37	78	1563	103	3600	3600
	III	43	318	16	25	309	39	78	1832	156	3600	3600
	IV	44	376	25	28	434	42	84	1409	142	3600	3600
	Average	47	317	18	25	372	40	82	1579	133	3600	3600
A3	I	249	514	324	85	2125	169	187	10174	857	3600	3600
	II	221	497	51	73	1947	132	102	11073	508	3600	3600
	III	191	509	350	73	1461	145	290	11013	978	3600	3600
	IV	189	507	125	84	2213	160	176	11760	605	3600	3600
	Average	213	507	213	79	1937	152	189	11005	737	3600	3600

[a]= estimated time (Vadseth et al., 2022).

solver in the groups with larger instances indicates very small average gaps, highlighting the quality of the solutions determined for these instances. In the set of instances with 30 customers, we achieved even better results, evidencing the technique's robustness and indicating promising perspectives for its application in real problems involving even larger dimensions. Despite this, some adjustments to optimize its computational performance are necessary to increase its effectiveness in instances with more than 50 customers. Given the excellent results obtained for the single-item problem, an alternative worth exploring is the incorporation of other metaheuristics or even exact methods that can be combined with decomposition strategies. Therefore, we believe that we have contributed to research in the area by presenting a technique that, even though little explored in the context of the PRP due to its historical relevance in the area of optimization, has showed efficient and robust in this research's computational experiments.

CRediT authorship contribution statement

Luiz Fernando Rodrigues: Conceptualization, Methodology, Software, Validation, Formal Analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Maristela Oliveira Dos Santos:** Conceptualization, Methodology, Validation, Investigation, Writing – original draft, Writing – review & editing, Supervision, Project Administration, Funding acquisition. **Bernardo Almada-Lobo:** Conceptualization, Methodology, Validation, Writing – review & editing, Supervision, Project Administration, Investigation.

Data availability

Data will be made available on request.

Table A.14
Overview of the Archetti et al. (2011) instances (Adulyasak et al., 2014).

Instances	A1	A2	A3
Number of instances	480	480	480
Number of periods	6	6	6
Number of customers	14	50	100
Number of vehicles	1	Unlimited	Unlimited
Demand	Constant	Constant	Constant
Production capacity	Unlimited	Unlimited	Unlimited
Plant inventory capacity	Unlimited	Unlimited	Unlimited
Customers inventory capacity	Constant	Constant	Constant
Initial inventory at the plant	0	0	0
Initial inventory at the customers	Variable	Variable	Variable
Vehicle capacity	Constant	Constant	Constant

Table A.15
Descriptions of the Archetti et al. (2011) classes (Adulyasak et al., 2014).

Class	Type	Descriptions
Class I	1–24	Standard instances
Class II	25–48	High production costs (Class I × 10)
Class III	49–72	Large transportation costs (Class I × 5)
Class IV	73–96	No customer inventory costs

Acknowledgments

This work was supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) under Grant 307466/2021-3; and it was carried out using the computational resources of the Center for Mathematical Sciences Applied to Industry (CeMEAI) funded by FAPESP under Grant 2013/07375-0.

Appendix A. Descriptions of the single item instances

To compare the MA proposed herein with other techniques in the literature, we will use the instances proposed by Archetti et al. (2011), as shown in Table A.14. Group A1 has only one vehicle and 14 customers, groups A2 and A3 have 50 and 100 vehicles, respectively, and vehicles are unlimited. The demands are constant and customers' initial inventories are variable, while customers' inventory capacity is limited. The production and inventory capacities at the plant are limited, but the initial inventories are zero. It should be noted that the instances are divided into four classes, according to Table A.15. Class I (1 to 24) has the basic cost configuration, Class II (25 to 48) has high production costs, Class III (49 to 72) has high transport costs and, finally, Class IV (73 to 96) has no inventory costs. Thus, each class has 24 instances with 5 seeds each, totaling 480 instances in this set.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cie.2023.109388.

References

Absi, N., Archetti, C., Dauzère-Pérès, S., & Feillet, D. (2015). A two-phase iterative heuristic approach for the production routing problem. *Transportation Science*, *49*, 721–1005. http://dx.doi.org/10.1287/trsc.2014.0523.

Absi, N., Archetti, C., Dauzère-Pérès, S., Feillet, D., & Speranza, M. G. (2018). Comparing sequential and integrated approaches for the production routing problem. *European Journal of Operational Research*, *269*(2), 633–646. http://dx.doi.org/10.1016/j.ejor.2018.01.052.

Adulyasak, Y., Cordeau, J.-F., & Jans, R. (2014). Optimization-based adaptive large neighborhood search for the production routing problem. *Transportation Science*, *48*(1), 20–45. http://dx.doi.org/10.1287/trsc.1120.0443.

Adulyasak, Y., Cordeau, J. F., & Jans, R. (2015). The production routing problem: A review of formulations and solution algorithms. *Computers & Operations Research*, *55*, 141–152. http://dx.doi.org/10.1016/j.cor.2014.01.011, arXiv:CIRRELT-2013-49.

Archetti, C., Bertazzi, L., Paletta, G., & Speranza, G. (2011). Analysis of the maximum level policy in a production-distribution system. *Computers & Operations Research*, *38*(12), 1731–1746. http://dx.doi.org/10.1016/j.cor.2011.03.002.

Armentano, V. A., Shiguemoto, A. L., & Løkketangen, A. (2011). Tabu search with path relinking for an integrated productiondistribution problem. *Computers & Operations Research*, *38*(8), 1199–1209. http://dx.doi.org/10.1016/j.cor.2010.10.026.

Avci, M., & Yildiz, S. T. (2019). A matheuristic solution approach for the production routing problem with visit spacing policy. *European Journal of Operational Research*, *279*(2), 572–588. http://dx.doi.org/10.1016/j.ejor.2019.05.021.

Berretta, R., & Moscato, P. (1999). The number partitioning problem: An open challenge for evolutionary computational? In D. Corne, M. Dorigo, F. Glover, D. Dasgupta, P. Moscato, R. Poli, & K. V. Price (Eds.), *New ideas in optimisation* (pp. 261–278). McGraw-Hill.

Berretta, R., & Rodrigues, L. F. (2004). A memetic algorithm for a multistage capacitated lot-sizing problem. *International Journal of Production Economics*, *87*(1), 67–81. http://dx.doi.org/10.1016/S0925-5273(03)00093-8.

Boudia, M., Louly, M. A., & Prins, C. (2007). A reactive GRASP and path relinking for a combined production-distribution problem. *Computers & Operations Research*, *34*(11), 3402–3419. http://dx.doi.org/10.1016/j.cor.2006.02.005.

Boudia, M., & Prins, C. (2009). A memetic algorithm with dynamic population management for an integrated production-distribution problem. *European Journal of Operational Research*, *195*(3), 703–715. http://dx.doi.org/10.1016/j.ejor.2007.07.034.

Brahimi, N., & Aouam, T. (2016). Multi-item production routing problem with backordering: A MILP approach. *International Journal of Production Research*, *54*(4), 1076–1093. http://dx.doi.org/10.1080/00207543.2015.1047971.

Chandra, P., & Fisher, M. L. (1994). Coordination of production and distribution planning. *European Journal of Operational Research*, *72*, 503–517. http://dx.doi.org/10.1534/genetics.112.141861.

Chitsaz, M., Cordeau, J. F., & Jans, R. (2019). A unified decomposition matheuristic for assembly, production, and inventory routing. *INFORMS Journal on Computing*, *31*(1), 134–152. http://dx.doi.org/10.1287/ijoc.2018.0817.

Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, *12*, 568–581. http://dx.doi.org/10.1287/opre.12.4.568.

Cordeau, J. F., Gendreau, M., Laporte, G., Potvin, J. Y., & Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, *53*(5), 512–522. http://dx.doi.org/10.1057/palgrave.jors.2601319.

Darvish, M., Archetti, C., & Coelho, L. C. (2019). Trade-offs between environmental and economic performance in production and inventory-routing problems. *International Journal of Production Economics*, *217*(July 2017), 269–280. http://dx.doi.org/10.1016/j.ijpe.2018.08.020.

Darvish, M., Kidd, M. P., Coelho, L. C., & Renaud, J. (2021). Integrated production-distribution systems: Trends and perspectives. *Pesquisa Operacional*, *41*(spe), 246080–246081. http://dx.doi.org/10.1590/0101-7438.2021.041s1.00246080.

Davis, L. (1985). Applying adaptive algorithms to epistatic domains. In *Proceedings of the 9th international joint conference on artificial intelligence* (pp. 162–164). San Francisco, CA: Morgan Kaufmann Publishers Inc.

Dolan, E., & More, J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, *91*(2), 201–213. http://dx.doi.org/10.1007/s101070100263.

França, P. M., Mendes, A., & Moscato, P. (2001). A memetic algorithm for the total tardiness single machine scheduling problem. *European Journal of Operational Research*, *132*(1), 224–242. http://dx.doi.org/10.1016/S0377-2217(00)00140-5, URL https://www.sciencedirect.com/science/article/pii/S0377221700001405.

Fumero, F., & Vercellis, C. (1999). Synchronized development of production, inventory, and distribution schedules. *Transportation Science*, *33*(3), 330–340. http://dx.doi.org/10.1287/trsc.33.3.330.

Izadi, L., Ahmadizar, F., & Arkat, J. (2020). A hybrid genetic algorithm for integrated production and distribution scheduling problem with outsourcing allowed. *IJE Transactions B*, *33*(11), 2285–2298.

Kayé, B. K. B., Diaby, M., Koivogui, M., & Oumtanaga, S. (2021). A memetic algorithm for an external depot production routing problem. *Algorithms*, *14*(1), 1–24. http://dx.doi.org/10.3390/a14010027.

Li, Y., Chu, F., Chu, C., & Zhu, Z. (2019). An efficient three-level heuristic for the large-scaled multi-product production routing problem with outsourcing. *European Journal of Operational Research*, *272*(3), 914–927. http://dx.doi.org/10.1016/j.ejor.2018.07.018.

Manousakis, E. G., Kasapidis, G. A., Kiranoudis, C. T., & Zachariadis, E. E. (2022). An infeasible space exploring matheuristic for the production routing problem. *European Journal of Operational Research*, *298*(2), 478–495. http://dx.doi.org/10.1016/j.ejor.2021.05.037.

Mendes, A. S., Franca, P. M., & Moscato, P. (2002). Fitness landscapes for the total tardiness single machine scheduling problem. *Neural Network World*, *12*(2), 165–180.

Miranda, P. L., Cordeau, J. F., Ferreira, D., Jans, R., & Morabito, R. (2018). A decomposition heuristic for a rich production routing problem. *Computers & Operations Research*, *98*, 211–230. http://dx.doi.org/10.1016/j.cor.2018.05.004.

Miranda, P. L., Morabito, R., & Ferreira, D. (2018). Optimization model for a production, inventory, distribution and routing problem in small furniture companies. *Top*, *26*(1), 30–67. http://dx.doi.org/10.1007/s11750-017-0448-1.

Moons, S., Ramaekers, K., Caris, A., & Arda, Y. (2017). Integrating production scheduling and vehicle routing decisions at the operational decision level: A review and discussion. *Computers & Industrial Engineering*, *104*, 224–245. http://dx.doi.org/10.1016/j.cie.2016.12.010.

Moscato, P., & Norman, M. G. (1992). A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems. In *International conference on parallel computing and transputer applications, no. January* (pp. 177–186).

Mostafa, N. A., & Eltawil, A. B. (2015). The production-inventory-distribution-routing problem: An integrated formulation and solution framework. In *IEOM 2015 - 5th international conference on industrial engineering and operations management, proceeding*. http://dx.doi.org/10.1109/IEOM.2015.7093751.

Mostafa, N., & Eltawil, A. (2019). Using valid inequalities to solve the integrated production-inventory-distribution-routing problem. *International Journal of Operational Research*, *35*(4), 551. http://dx.doi.org/10.1504/ijor.2019.10022814.

Neves-Moreira, F., Almada-Lobo, B., Cordeau, J. F., Guimarães, L., & Jans, R. (2019). Solving a large multi-product production-routing problem with delivery time windows. *Omega (United Kingdom)*, *86*, 154–172. http://dx.doi.org/10.1016/j.omega.2018.07.006.

Qiu, Y., Qiao, J., & Pardalos, P. M. (2019). Optimal production, replenishment, delivery, routing and inventory management policies for products with perishable inventory. *Omega (United Kingdom)*, *82*, 193–204. http://dx.doi.org/10.1016/j.omega.2018.01.006.

Qiu, Y., Wang, L., Fang, X., Pardalos, P. M., & Goldengorin, B. (2018). Formulations and branch-and-cut algorithms for production routing problems with time windows. *Transportmetrica A: Transport Science*, *14*(8), 669–690. http://dx.doi.org/10.1080/23249935.2018.1427157.

Qiu, Y., Wang, L., Xu, X., Fang, X., & Pardalos, P. M. (2018a). A variable neighborhood search heuristic algorithm for production routing problems. *Applied Soft Computing*, *66*, 311–318. http://dx.doi.org/10.1016/j.asoc.2018.02.032.

Qiu, Y., Wang, L., Xu, X., Fang, X., & Pardalos, P. M. (2018b). Formulations and branch-and-cut algorithms for multi-product multi-vehicle production routing problems with startup cost. *Expert Systems with Applications*, *98*, 1–10. http://dx.doi.org/10.1016/j.eswa.2018.01.006.

Ramos, B., Alves, C., & Valério de Carvalho, J. (2020). An arc flow formulation to the multitrip production, inventory, distribution, and routing problem with time windows. *International Transactions in Operational Research*, *00*, 1–28. http://dx.doi.org/10.1111/itor.12765.

Ruokokoski, M., Solyali, O., Cordeau, J., Jans, R., & Sural, H. (2010). Efficient formulations and a branch-and-cut algorithm for a production-routing problem. *Transportation*, 1–43.

Russell, R. A. (2017). Mathematical programming heuristics for the production routing problem. *International Journal of Production Economics*, *193*(May), 40–49. http://dx.doi.org/10.1016/j.ijpe.2017.06.033.

Salehi Sarbijan, M., & Behnamian, J. (2020). Multi-product production routing problem by consideration of outsourcing and carbon emissions: Particle swarm optimization. *Engineering Optimization*, 1–17. http://dx.doi.org/10.1080/0305215X.2020.1786080.

Senoussi, A., Dauzère-Pérès, S., Brahimi, N., Penz, B., & Kinza Mouss, N. (2018). Heuristics based on genetic algorithms for the capacitated multi vehicle production distribution problem. *Computers & Operations Research*, *96*, 108–119. http://dx.doi.org/10.1016/j.cor.2018.04.010.

Solyalı, O., & Süral, H. (2017). A multi-phase heuristic for the production routing problem. *Computers & Operations Research*, *87*, 114–124. http://dx.doi.org/10.1016/j.cor.2017.06.007.

Toledo, C., França, P., Morabito, R., & Kimms, A. (2009). Multi-population genetic algorithm to solve the synchronized and integrated two-level lot sizing and scheduling problem. *International Journal of Production Research*, *47*(11), 3097–3119. http://dx.doi.org/10.1080/00207540701675833.

Vadseth, S. T., Andersson, H., & Stålhane, M. (2022). A multi-start route improving matheuristic for the production routing problem. *XX*.

Wagner, H. M., & Whitin, T. M. (1958). Dynamic version of the economic lot size model. *Management Science*, *5*(1), 89–96. http://dx.doi.org/10.1287/mnsc.5.1.89.

Yagmur, E., & Kesen, S. E. (2021). Multi-trip heterogeneous vehicle routing problem coordinated with production scheduling: Memetic algorithm and simulated annealing approaches. *Computers & Industrial Engineering*, *161*, Article 107649. http://dx.doi.org/10.1016/j.cie.2021.107649.

Zhang, Z., Luo, Z., Baldacci, R., & Lim, A. (2021). A benders decomposition approach for the multivehicle production routing problem with order-up-to-level policy. *Transportation Science*, *55*(1), 160–178. http://dx.doi.org/10.1287/TRSC.2019.0964.