A memetic algorithm for the inventory routing problem

Mohamed Salim Amri Sakhri1 · Mounira Tlili2 · Ouajdi Korbaa3

Received: 25 August 2021 / Revised: 20 December 2021 / Accepted: 30 March 2022 / Published online: 22 April 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
In this article, we study an Inventory Routing Problem with deterministic customer demand in a two-tier supply chain. The supply chain network consists of a supplier using a single vehicle with a given capacity to deliver a single product type to multiple customers. We are interested in population-based algorithms to solve our problem. A Memetic Algorithm (MA) is developed based on the Genetic Algorithm (GA) and Variable Neighborhood Search methods. The proposed meta-heuristics are tested on small and large reference benchmarks. The results of the MA are compared to those of the classical GA and to the optimal solutions in the literature. The comparison shows the efficiency of using MA and its ability to generate high quality solutions in a reasonable computation time.

Keywords Inventory routing problem · Genetic algorithm · Memetic algorithm · Variable neighborhood search · Order crossover · Optimization

1 Introduction
Transport activities are an important part of logistics, which is constantly evolving in response to the globalisation of trade, computerisation of processes, new consumption patterns and environmental requirements. Logistics managers coordinate the entire...
supply chain, including the various operations of order preparation, procurement, fleet management, transportation planning, warehousing and inventory management.

In this context, information and communication technologies are becoming increasingly important. In fact, they ensure integrated management between the different compartments of the distribution network, geolocation, product traceability and inventory management. Moreover, integrated management has created new challenges to be taken into account. New IT skills are needed to solve these different types of problems related to inventory, business relationships and eco-driving (reducing fuel consumption, minimizing inventory access time and minimizing inventory costs).

The Inventory Routing Problem (IRP) is a distribution problem that combines two supply chain activities: inventory management and vehicle routing. It aims to determine, at each period, which customer should be visited, the quantity of product to be delivered and the vehicle’s route. Solving this problem means minimizing the holding costs of the customer’s and supplier’s inventory and the transportation costs over the planning horizon. This type of problem is common in industry during the delivery and replenishment stages of a finished or semi-finished product or for spare parts.

IRP was first introduced by Bell et al. (1983). They considered IRP as a process of distributing a single product over several time periods from a supplier to a set of customers, following a constant (deterministic) demand over time, with an infinite and homogeneous fleet of vehicles. Since its appearance, many variants of IRP have been proposed in the literature.

In this work, we focus on research that has considered a periodic IRP with a deterministic query, as in the work of Qin et al. (2014). Known variants of IRP in this line of research that have treated the single product case as in Amri Sakhri (2021) or the multi-product case studied in Meysam et al. (2021). Most of the researchers have studied IRP with a single vehicle fleet or multiple vehicles by Bertazzi and Speranza (2002) and Mjirda et al. (2014) respectively. In the case of multi-vehicle IRP, the types of vehicle fleet that can be considered are either homogeneous as in Olgun and Aydemir (2021) or heterogeneous as in Jahangir et al. (2019). The proposed replenishment policies for the different IRPs are Order-up-to level (OU), Maximum-level (ML) or Optimized Target Level (OTL) which are respectively studied by Bertazzi et al. (2002), Archetti et al. (2007) and Coelho and Laporte (2015).

IRP solution methods are mainly classified into two categories: exact methods and approximate methods. An exact method is a solution method that guarantees to obtain an optimal solution to a given optimization problem. In terms of solving IRPs, the best-known exact methods are the Branch-and-Cut algorithm, the Branch-and-Price algorithm, and the Branch-and-Price-and-Cut algorithm, used respectively by Eleftherios et al. (2021), Hewitt et al. (2013) and Gizem et al. (2021). Approximate methods consist in finding a satisfactory solution to an optimization problem in a reasonable computation time, especially when the search area of the problem is large and complex. This type of methods has been successfully applied in solving different variants of IRP and in particular the meta-heuristics which are the most used approaches in this field. The most used meta-heuristics in solving the variants of the IRP are the Variable Neighborhood Search (VNS), the Tabu Search (TS), the Genetic Algorithm (GA), the Ant Colony Algorithm (ACA) and the Particle Swarm Optimization (PSO), applied
respectively by Gruler et al. (2018), Mahdi et al. (2021), Azadeh et al. (2017), Fardi et al. (2019) and Rau et al. (2018).

In this article, we consider one of the most basic and well-studied IRP under the Vendor Managed Inventory (VMI) system. This problem is introduced and solved by an exact method in Archetti et al. (2007). In our study, we will not only consider small instances of Archetti et al. (2007) but also large test instances of Archetti et al. (2012). Solving such instances requires the use of approximate methods to allow results in acceptable computation times, which is a limitation for exact methods.

Efficient approaches have been proposed to solve the same model, such as the meta-heuristic of Archetti et al. (2012) called Hybrid Approach to Inventory Routing (HAIR). HAIR is a powerful local search approach that has yielded acceptable results for solving the Archetti et al. (2007) model. This method combines a TS scheme with ad hoc designed mixed integer programming models. Another recent population-based search method was implemented by Amri Sakhri et al. (2018) using the Generic Algorithm (GA). This method is characterized by the diversification of its solutions generated by creating populations in a random way and a random reproduction made by applying genetic operators. However, this method was not aggressive enough for combinatorial optimization problems compared to other local search meta-heuristics due to its randomness.

Memetic Algorithms (MAs) are more powerful versions of GAs that apply a local search procedure to each new solution to improve it. The use of MAs will allow us to improve the performance of the GA thanks to an intensification phase performed by the local search heuristic that we will introduce. These reasons motivated us to develop a flexible MA, capable of tackling our goal, while being competitive with already published methods. MA has often been used to solve the Vehicle Routing Problem (VRP) and has been shown to be effective in producing high quality results as in Sabar et al. (2020), Molina et al. (2020) and, Ece and Saadettin (2021). Note that no one has yet tried to apply the MA for IRP, which prompted us to do so.

The main contributions of this paper can be summarized in the following points:

- Development of a GA that uses better crossover restructuring, random mutation and a quality selection mechanism, which promote better diversity for the GA application to the resolution of a well-known IRP.
- A time-varying fitness function that simultaneously determines the cost of vehicle route and inventory storage in the replenishment network and improves search performance.
- An evolutionary memetic approach that uses genetic operators and a powerful local search method, which is VNS to more efficiently solve the proposed IRP.
- Stop criteria based on the number of iterations for simulating a hybrid algorithm should be evaluated chronologically to determine a more efficient computation time result.
- The two algorithms developed were tested on IRP reference instances and their results are compared with the methods proposed in the literature.
- The good performance of MA has been demonstrated.

The rest of the article is organized as follows. In the next section, we formally describe our problem, and present a mathematical programming formulation for IRP.
The third section presents the different algorithms developed to solve the proposed problem. In the fourth section, we present the characteristics of the benchmarks used, the way our parameters are calibrated and analyse in depth the results of the calculation experiments. In the last section, we conclude our work and give some perspectives for future research.

2 IRP model

In this work, we focus on the study, analysis and solution of an NP-Hard problem, which is IRP. Our objective is to optimize the total supply chain costs and avoid stock-outs for customers. In this section, we will describe the mathematical model to understand the solution structure of our problem. The considered IRP can handle four problems: customer and supplier inventory management, stock quantities to be delivered to avoid stock-outs, customer delivery period assignment, vehicle route design and optimization.

2.1 Problem description

The IRP can be defined as a Mixed Integer Linear Programming (MILP), in which the supplier has to make concurrent decisions, which serve to optimally organize inventory levels at customers and transfer precise quantities of stock from its site to the selected customers during each delivery period. The problem can be represented by a graph consisting of a set of nodes $N = \{0, \ldots, n\}$, where the node 0 is the supplier, and the subset $N' = N/\{0\}$ represents the customers. $C_{ij}$ is the cost associated to the route between nodes i and j. For each node $i \in N$, the inventory holding cost is $h_i$ per unit and per period. At each period $t \in T$, a quantity of products $p^t$ is added to the available supplier stock and a quantity of stock $r^t_i$ is consumed by the customer $i \in N'$.

The binary variable $Y^t_{ij}$ is equal to 1 if the edge $(i, j) \in N$, otherwise it is equal to 0. The binary variable $x^t_i$ is equal to 1 if node $i \in N$ is visited at period $t \in T$, otherwise it is equal to 0. The variables $I^t_i$ is used to indicate the inventory level at node $i \in N$ at the end of each replenishment period $t$. The supplier and the customers have a predefined initial inventory level equal respectively to $I^0_0$ and $I^0_i$. For each customer $i \in N'$ there is a maximum and minimum level of inventory denoted by $U_i$ and $L_i$. In our problem, only one vehicle of capacity denoted by $Q$ is used. The variable $q^t_i$ is the quantity delivered to a customer $i$ at period $t \in T$. The variable $v^t_i$ is the quantity of stock that the vehicle carries, before the inventory is replenished at customer $i$ at each period $t \in T$.

2.2 Assumptions

- The production capacity of the supplier is limited but sufficient to satisfy the consumers’ demands.
- At the end of each period, the total quantity of inventory in the different nodes of the network is measured to determine the total inventory cost.
Customer inventory capacities are limited and shortages are not allowed. The demands of the customers could not be delivered partially.

### 2.3 Mathematical modeling of IRP

The mathematical model is inspired by Archetti et al. (2007). The objective function is to minimize the total operational costs, formulated as follows.

\[
\min \sum_{t} h_{0} I_{0}^{t} + \sum_{t} \sum_{i} h_{i} I_{i}^{t} + \sum_{t} \sum_{i} \sum_{j} C_{ij} Y_{ij}^{t} \quad (1)
\]

These operational costs represent the sum of the inventory costs at the supplier and at the customers, plus the costs of the routes over the time horizon. The constraints of this model will be used in the next section to develop our method of resolution. To better understand the model, we will treat the constraints in separate subsections, each of which is composed of constraints with the same functionality.

#### 2.3.1 Inventory level constraints

The constraint of inventory level at the supplier at the period \( t \):

\[
I_{0}^{t} = I_{0}^{t-1} + p' - \sum_{i} q_{i}^{t} \quad \forall \ t \in T \quad (2)
\]

The constraint of inventory level at the customers at the period \( t \):

\[
I_{i}^{t} = I_{i}^{t-1} - r_{i}^{t} + q_{i}^{t} \quad \forall \ i \in N', \forall \ t \in T \quad (3)
\]

Constraints enforce the inventory level to stay between lower bound and upper bound:

\[
I_{i}^{t} \geq L_{i} \quad \forall \ i \in N', \forall \ t \in T \quad (4)
\]

\[
U_{i} \geq I_{i}^{t} \quad \forall \ i \in N', \forall \ t \in T \quad (5)
\]

#### 2.3.2 Replenishment constraints

The following constraints ensure the requirements of the Order-Up-to level (OU) policy. This policy imposes that if a customer is visited, the quantity delivered is such that the maximum inventory level is reached:

\[
q_{i}^{t} \geq U_{i} x_{i}^{t} - I_{i}^{t-1} \quad \forall \ i \in N', \forall \ t \in T \quad (6)
\]

\[
q_{i}^{t} \leq U_{i} - I_{i}^{t-1} \quad \forall \ i \in N', \forall \ t \in T \quad (7)
\]

\[
q_{i}^{t} \leq U_{i} x_{i}^{t} \quad \forall \ i \in N', \forall \ t \in T \quad (8)
\]
2.3.3 Vehicle routing constraints

The flow conservation constraint:

\[ \sum_{i}^{N} Y_{ij}^{t} = \sum_{i}^{N} Y_{ji}^{t} \quad \forall \ j \in N', \ i \neq j, \ \forall \ t \in T \] (9)

The vehicle capacity constraint:

\[ \sum_{i}^{N'} q_{i}^{t} \leq Qx_{i}^{t} \quad \forall \ t \in T \] (10)

Sub-tour elimination constraints for each vehicle route at each period:

\[ v_{j}^{t} - v_{i}^{t} + QY_{ij}^{t} \leq Q - q_{i}^{t} \quad \forall \ i, j \in N', \ i \neq j, \ \forall \ t \in T \] (11)
\[ q_{i}^{t} \leq v_{i}^{t} \leq Q \quad \forall \ i \in N', \ \forall \ t \in T \] (12)

2.3.4 Non-negativity and integrality constraints

\[ Y_{ij}^{t} \in \{0, 1\} \quad \forall \ i, j \in N', \ i \neq j, \ \forall \ t \in T \] (13)
\[ x_{i}^{t} \in \{0, 1\} \quad \forall \ i \in N', \ \forall \ t \in T \] (14)
\[ I_{i}^{t} \geq 0 \quad \forall \ i \in N, \ \forall \ t \in T \] (15)
\[ q_{i}^{t} \geq 0 \quad \forall \ i \in N', \ \forall \ t \in T \] (16)
\[ v_{i}^{t} \geq 0 \quad \forall \ i \in N', \ \forall \ t \in T \] (17)

2.3.5 The usefulness of constraints

The objective function (1) will be calculated from the fitness of GA and MA. The customer inventory level constraint (3) through (5) will be used to design the sorting program that will allow us to identify the customers assigned to each replenishment period. The OU inequalities (6) through (8) will be used to design the program to compute the optimal quantities of inventory to transfer to the selected customers at each delivery period. The vehicle routing constraints (9) through (12) will be used to define the initial GA and MA solutions. The inventory level constraints (2) and (3) are updated at the end of each period to prepare for the next replenishment cycle. Constraints (13) through (17) will ensure the feasibility of the model.

3 Meta-heuristic description

Solving an NP-Hard optimization problem with an exact method will limit our field of action because these methods lose their efficiency when the problem size increases.
This led us to choose the family of evolutionary algorithms for solving our IRP. These algorithms can handle combinatorial problems with a large number of mixed variables.

The developed meta-heuristic consists of two optimization phases. In the first phase, the sorting algorithm tries to find all the customers to be served at each period while respecting the delivery priority constraints. The algorithm also defines the quantities to be delivered while respecting the constraints of vehicle capacity, the quantity of stock available at the supplier and the customer’s admissible stock level. It ends up generating a first non-optimized solution (a route).

The second phase consists in optimizing the initial solution obtained using the genetic operators and the local search method described later. In this phase, the path that minimizes the distance traveled by the vehicle to meet the customer’s request without violating the vehicle’s routing constraints is sought.

In this study, two meta-heuristics are proposed to solve the IRP model. The first one is the GA, composed of the two-point crossover using the Order Crossover (OX) technique and the classical mutation operator which operates randomly. The general structure of the GA developed is described in Fig. 1.

---

Fig. 1  Flowchart of Genetic Algorithm (GA)
Figure 1 shows the GA application process. First, GA generates an initial population composed of a number of routes while respecting the VRP constraints mentioned in the previous model. Then, GA proceeds to the exploration of the search space using crossover and mutation operators. The goal of these genetic operators is to enrich the diversity of the population by manipulating the structure of the chromosomes (routes).

The second meta-heuristic is MA. This algorithm keeps the same GA steps by adding a local search operator, which is the VNS after the mutation operator. The VNS operator is used to improve the results of our computational tests. The VNS adds search intensification to the diversification provided by the GA, i.e., each search space will be explored in depth to find its local optimum. This process ensures a better convergence speed of the algorithm to an optimal solution. Then, a new population of heterogeneous individuals is established by the selection operator. Finally, GA/MA runs on a fixed number of iterations and the best fitness score of the individuals is considered as the solution of the selected instance. The optimization process of each meta-heuristic is described in the following subsections.

3.1 General structure of genetic algorithm

In this subsection, we will describe the functional process of GA and introduce its different operators. Then, we will introduce the local search operator (VNS) that will transform our GA into MA.

3.1.1 Construction of an initial solution

The main objective of the building phase is to determine the set of customers that are likely to be out of stock during each period and those that minimize supply chain costs. This step considers both the inventory level and the cost of storage at the customers. The constraints that can be considered are the stock level equation and the minimum stock level inequality, and we also need the supplier and customer storage cost parameter.

The choice of the initial population is important because it influences the speed of convergence of the algorithms to the global optimum. The construction of the initial solution controls the amount of stock (transferred and on hand), at the different nodes of the distribution network, in order to avoid stock-outs at each period and to minimize the total storage costs. The control of the transported stock results from the vehicle capacity constraint and the quantity to be delivered which is determined by the Orde-up-to-level inequalities.

3.1.2 Assignment of customers at each period

This step will identify the set of customers to be served at each period. The initial population contains an initial unoptimized composition of vehicle routes (individuals). In this level, the algorithm developed makes it possible to classify the customers in ascending order according to the average number of periods necessary to consume the quantity $U_i - L_i$ for each customer and the customers with the same delivery period are classified in descending order according to their storage cost. When a customer $i$ is considered in this population, his delivery period is determined.
In the initialization phase of the GA, a feasible solution to the problem is constructed which contains the customers presenting an immediate stock-out risk, the constraints considered at this level are those of the inventory level at period t as well as the minimum inventory level at the same customer. If the vehicle capacity constraint is not violated, customers whose storage cost is lower than that of the supplier are selected and added to the initial individual. Figure 2 shows the process of ranking the customers according to the parameters described below with $L_i = 0$ for all network customers.

### 3.1.3 The generation of the initial population

This step aims at generating an initial population of P chromosomes, each chromosome being composed of two individuals. This population represents the set of delivery
rounds serving as a basis for future generations. The set of individuals is generated randomly, each individual is composed of customers selected by the ranking algorithm we just saw in the previous paragraph.

### 3.1.4 Evaluation function

The function for evaluating an individual is the most sensitive part of our algorithm. This function uses all the components of the model in a simulation that yields a result known as the chromosome fitness value. This result is evaluated during the rest of the GA steps to determine a chromosome structure with an improved fitness value. The chromosome fitness value is determined by the vehicle route costs and the inventory holding costs at the various nodes in the network.

The quality of an individual is reflected in the fitness function. The quality of the generated result is deduced by comparing this result to the best initial solution. The fitness of the individuals must be determined, as it is needed for the selection and replacement steps.

This function has two main objectives. The first is to find the shortest delivery path that starts from a supplier, passes through the different customers of the replenishment network and ends at the same starting supplier. The second objective is to satisfy the demand of the visited customers at each period in order to replenish them with the stock quantities determined by the order up to level inequalities under the vehicle capacity constraint.

### 3.1.5 The selection operator

This operator is used to evaluate the individuals who are most likely to achieve the best results. In order to ensure that all individuals have an equal chance of being chosen and included in the selection process, we choose the uniform selection operator. The principle of this operator is such that the selection is done randomly, uniformly and without intervention of the adaptive value. Thus, each individual has a probability of $1/N$ of being selected, where $N$ is the total number of individuals in the population.

### 3.2 Improving the delivery routes

At this level, we will use the skills of our GA to improve the initial solution. This step aims at optimizing only the distance traveled by the vehicle without taking into account the other objectives defined in the previous step. Three genetic operators will be used here which are the crossover, mutation and replacement operators. Then, we will transform GA into MA by adding VNS after the mutation operator of GA. In the following, we will describe the execution process of each operator.

#### 3.2.1 Crossover

Chromosomes can undergo different types of crossover (one point crossover, two-point crossover and uniform crossover). The two-point crossover is the best suited for cyclic
permutations. The way to restructure the chromosomes after the crossover process is also effective in covering a search space more quickly. In our case, we opted for the Order Crossover Operator (OX) which has shown its efficiency in different case of routing problems as in Prins (2004) and Labadi et al. (2008). This type of crossover takes place as follows.

A two-point crossover is made between the first and the last gene of the chromosome. The genes are coded with integers. The positions of the two cut points are chosen randomly. First, the genes between the cut points are copied in the same order to the same position in the offspring. Then, starting from the second cut point of one parent, the genes of the other parent are copied in the same order, omitting the existing genes.

The example in the Fig. 3 shows the OX process to build the first offspring. We start by performing the exchange of gene sequences between the cut points of P1 and P2. The sequence of the genes of the first parent from the second cut point is 5-7-6-1-3-4-2-8. After deleting genes 7, 1, 4 and 2, which are duplicated in the first child, the new sequence that will be copied from the second cut point is 5-6-3-8. The same process is applied for the second child.

3.2.2 Mutation

Mutation is considered the operator responsible for maintaining genetic diversity in a population. The mutation operator provides the ergodicity property of space exploration by GAs. This property indicates that the GA can reach all points in the search space without iterating through all these points in the resolution process. The convergence properties of GAs are therefore theoretically strongly dependent on this operator. The performance of the GA will be improved by applying at each iteration a mutation operation at each new generation, in which two genes of the same individual will

Fig. 3 Order Crossover OX process
be randomly chosen and their values exchanged. Thus, in our application, a random position exchange is performed between two customers.

### 3.2.3 The replacement operator

This operator is used to identify the best offspring obtained from the successive application of the selection, crossover and mutation operators and to eliminate the bad ones. It allows to reintroduce in the new population the individuals of the parent and the offspring with the best fitness value. Before inserting them in the new population, the individuals must be evaluated according to their fitness to choose the most suitable ones that reduce the total cost of the network.

In our work, we choose the tournament technique as the selection operator. This technique uses a random selection of parent pairs. Then, they will be compared to their offspring according to their fitness. Finally, we will choose the pair of individuals with the highest fitness score. The way this operator works is described in the following Fig. 4.

According to Fig. 4, we select among all pairs of chromosomes (P1, P2, C1 and C2) the chromosomes that generate the lowest network costs using a ranking function. It is possible that some individuals participate in several tournaments if their fitness score allows them to win several times, thus favoring the survival of their genes.

### 3.2.4 Stop condition

The algorithm stops when a predefined number of iterations is reached.

---

**Fig. 4** Selection process of GA
3.3 Memetic operator

The selection processes presented in the previous subsection are very sensitive to differences in fitness. In some cases, a very good individual may be reproduced too often. This may even lead to the complete elimination of its conspecifics, resulting in a homogeneous population containing only one type of individual. Thus, this individual can be the only representative of the following generations after many successive calculation tests.

To avoid this behavior, there are local search methods that optimize searches in each space traversed by the GA. To benefit from the advantages of these methods, we will use a powerful hybrid GA known as the Memory Algorithm (MA). Indeed, only one modification of the GA structure will be performed. A local search method, which is VNS, will be introduced after the mutation is applied. In this subsection, we will present the VNS hybridization operator that will allow our GA to improve its performance.

3.3.1 Variable neighbourhood search (VNS)

The great strength of GAs is their ability to cover the area of the solution space containing the different local optima of the function. However, they are inefficient when it comes to finding the exact value of the global optimum of this area. This is precisely what local optimization algorithms do best. Therefore, it is natural to think of combining a local search algorithm with GA to find the exact values of the optimum.

This optimization can easily be achieved by applying a local search method to the best paths found by GA. In fact, the association GA-VNS is a quasi necessity, the two methods are complementary and have different fields of application. GA eliminates the combinatorial aspect of the problem. This allows the local methods to find the best solution in each of the explored areas that can contain the global optimum.

Here, we introduce a powerful local search method that follows the random mutation process presented earlier. In this step, our algorithm will opt for customer position changes that generate a least-cost path. To do this, we will use the VNS method developed in Mladenovic and Hansen (1997). VNS is based on a path change process, favoring a diversification phase that aims to escape the local minimum.

This local search method can be viewed as a general framework for managing the execution order of local search procedures. A VNS methodically uses several types of neighborhoods. Its basic idea is to perform an exchange in two phases: a local optimum is obtained during the Local Search Phase and a perturbation of the solution is performed during the Shaking Phase. We will develop these different phases of the VNS in the following.

3.3.2 The VNS strategy for fitness improvement

The application of VNS as a local search method in MAs for solving VRP problems is rare, despite its effectiveness which has been proven by some researchers, as in Sbai et al. (2020). In this section, we will describe the VNS process applied to a vehicle routing case.
We consider an initial solution $s$.

- Shaking: We generate a neighbouring solution $s_1 \in M_1(s)$, from the set of neighbourhood structures $M_1(s)$, it presents the list of customers in the neighbourhoods of each customer.
- Local Search: We applied a local search procedure to obtain a solution $s_2$.
- If $f(s_2) < f(s)$, with $f(s)$ presents a part of our objective function related to the transport cost, that means the neighbourhood generates a lower cost path that improves the solution $s$,
- Then, we set $s = s_2$: we select the list (L) of customers of this neighbourhood, and we repeat the process with the first neighbourhood of L generating a new neighbouring solution in $M_1(s_2)$.
- Otherwise, we change the neighbourhood and repeat the process: the current solution remains $s$ and change neighbourhood by generating a solution $s_1$ in $M_2(s)$.

This process is described in the Fig. 5.

The following example in Fig. 6 shows how to create the first child $C_1^*$ with the VNS process. First, we randomly select the first node $i$. Then, we change the position of its successor node (next node) with the first node from the predefined neighborhood list for the selected node. If changing the position of the nodes improves the cost function of the route, then we make this change. Otherwise, we will randomly select another node from the list and repeat the process.

This process is repeated until a new node is found that reduces the total route cost. In the example in Fig. 6, the code of the randomly selected customer is 4, in $C_1$. If we need to test all the customers in the neighborhood of this customer, the code of the successor customer that reduces the total cost of the route will be 3, the switch will be between nodes 2 and 3, and the new structure of $C_1$ will be 0-1-4-3-2-0.
By adding the local search operator (VNS), we seek to improve the GA solutions. This will be the challenge of our upcoming computational tests.

4 Experimentation

4.1 Implementation and benchmarks

The developed GAs and MAs were implemented in Java EE-like for the ECLIPSE environment and run on a PC with an Intel Core i7-5500U @ 2.40 GHz processor and 8 GB of RAM. We applied our algorithm to the set of small instances proposed in Archetti et al. (2007), and then to the large instances given in Archetti et al. (2012) from 50 to 200 customers. The characteristics of these instances are described, briefly, as follows:

– Time horizon \( T = \{3; 6\} \);
– Number of customers: we will be interested in two sets of instances. The first one contains 50 small instances of the model up to 50 customers when \( T = 3 \) and 60 instances up to 200 customers composed of small and large instances when \( T = 6 \);
– Product quantity \( r_{ti} \) consumed by customer \( i \) at each period \( t \) : randomly generated as an integer number in the interval \([10, 100]\);
– Product quantity \( p_{t} \) made available at the supplier at each period \( t \) : \( \sum_{N}^{'i} r_{ti} \);
– Maximum inventory level \( U_{i} \) at customer \( i \) : it is given by \( r_{ti} \ast g_{i} \) where \( g_{i} \) is selected randomly from \((2, 3)\) and represents the number of time units needed to consume the quantity \( U_{i} \);
– Starting inventory level \( I^{'0}_0 \) at the supplier : \( \sum_{N}^{'i} U_{i} \);
– Starting inventory level \( I^{'0}_i \) at the customer \( i \) : \( U_{i} - r_{ti} \);
– Inventory cost at customer \( i \in N' \) is \( h_{i} \) : randomly generated in the intervals \([0.01, 0.05]\);
– Inventory cost at the supplier \( h_0 = 0.03 \);
– The number of vehicles is \( v = 1 \) and its capacity is given by: \( 3/2 \star \sum \dot{r}_i \);
– Transportation cost \( C_{ij} : \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \), where the points \((x_i, y_i)\) and \((x_j, y_j)\) are obtained by randomly generating each coordinate as an integer number in the interval \([0, 500]\);
– Replenishment policy: The OU policy, according to which every visit to a customer brings its inventory to the maximum level (ML), more details about the OU policy in Bertazzi et al. (2002).

4.2 Parameter setting

In population-based meta-heuristics, certain parameters must be initialized before the algorithm is executed. An appropriate initial setting of the parameters has an important impact on the progress of the solution, such as the rate of exploitation or exploration of the search space, and thus on the quality of the solution, Cooray and Thashika (2017). This parameter setting is done according to the problem to be solved. This practice is called parameter tuning. A good calibration of the parameters can lead to a reduction of the computation time and to an improvement of the quality of the search process, Baniamerian et al. (2019). In this paper, the parameters considered for calibration are the crossover, mutation, and VNS probabilities. In addition, the number of individuals in the initial population and the number of iterations are fixed according to the problem size.

In our choices for the calibration of the parameters, we have based ourselves on the literature. The experiments we carried out with the two meta-heuristics (GA-OX and MA) are performed with the same conditions and parameters. The probability of crossover \( P_c \) is 0.5 and the probability of mutation \( P_m \) is 0.1. These probabilities are general and natural, inspired by biological studies and adapted by several researchers such as Prasetyo et al. (2018) whose choice of \( P_c \) can belong to the interval \([0.5; 0.8]\) and in most IRP studies such as Park et al. (2016) and Abid et al. (2017) the \( P_m \) is equal to 0.1. For MA, we add the probability of VNS, which is \( P_{\text{vns}} = 0.1 \) a probability similar to that of \( P_m \) to maintain the randomness aspect of the hybrid GA. The initial population is equal to 50 paths for small instances in Archetti et al. (2007) and equal to 100 paths for large instances in Archetti et al. (2012).

For the number of iterations, we randomly chose instances in each benchmark and initially set the computational time as a stopping criterion to 1 h for the instances in the benchmarks of Archetti et al. (2007) and 12 h for those in Archetti et al. (2012). Then, we studied the number of iterations needed to obtain the final results for each tested instance and replaced the time stopping criterion with the number of iterations stopping criterion. This process allowed us to be more efficient and to reduce the CPU. Indeed, the number of iterations performed to obtain convergent solutions is about 150 iterations, for small instances up to 20 customers with \( T \in \{3; 6\} \) and the number of iterations increases for the rest of the small instances but remains below 350. For large instances up to 200 customers, about 900 iterations are needed to obtain a converged solution.
For safety reasons, we increased the number of iterations and set the maximum number of iterations to 200 for small instances up to 20 customers for $T \in \{3; 6\}$ and 400 iterations for the rest of the small instances. For the benchmark of Archetti et al. (2012), the maximum number of iterations is equal to 1000. Indeed, increasing these values would increase the computation time, but it could lead to a better solution. Note that, each instance of Archetti et al. (2007) is executed 30 times, the average results of these executions are considered as the solution of the executed instance.

4.3 Computational results

In this subsection, we will test the performance of the developed GA and MA applied on the benchmarks of Archetti et al. (2007) and Archetti et al. (2012). The chosen problem was initially solved by the B&C algorithm of the Cplex solver, in Archetti et al. (2007). The mentioned method showed its limitations in terms of result and computation time by increasing the problem size. For instances with three delivery periods, the number of customers considered should not exceed fifty. By spreading the planning horizon over six delivery periods, the number of customers is reduced to thirty. The exact algorithm loses its ability to solve the problem in a reasonable computation time by increasing the complexity of the problem. To deal with larger benchmarks like (Archetti et al. 2012), it is useful to use heuristic methods, but these methods must prove their ability to solve this type of problem first on small instances before testing them on large instances. This work will be the subject of the next paragraph.

Table 1 shows the results obtained by our developed meta-heuristics and those of previous research, tested on the same benchmark of Archetti et al. (2007). The first column shows the size of the sets of instances. The second column shows the optimal solution of the problem solved on Archetti et al. (2007) with the B&C algorithm, considered by the following methods as the lower bound (LB). The third, fourth and sixth columns are, respectively, the average of the optimal results obtained with HAIR from Archetti et al. (2012), GA-OX and MA developed in this paper. The fifth and seventh columns are, respectively, the average computational time required to obtain the GA-OX and MA results. The last column gives the upper bound (UB) of each set of instances determined by Archetti et al. (2007). Note that the UB is a limit beyond which the result becomes unacceptable.

The results of our tests and those of researchers who considered small instances of Archetti et al. (2007), are displayed in Table 1. As a first observation, we note that the developed algorithms respected the solution intervals determined by Archetti et al. (2007) for all sets of instances. This leads us to investigate further and see the benefits of using these two algorithms. We will start by comparing the results of GA to those of MA. This will allow us to identify which of the two algorithms is more effective in solving the problem at hand. Next, we will compare the results of our best algorithm to those of HAIR. Finally, we will determine the error rate of these algorithms based on the exact solution of the problem.

GA was able to accurately solve four of the ten instance sets with a planning horizon of three periods, and two of the six instance sets with a planning horizon of six periods.
Table 1  Average result values obtained by the different resolution methods for the small-instances

| Instance | LB      | HAIR    | GA-OX   | Time (s) | MA      | Time (s) | BS     |
|----------|---------|---------|---------|----------|---------|----------|--------|
| T = 3    |         |         |         |          |         |          |        |
| Small-5  | 1418.76 | 1418.76 | 1418.76 | 71.95    | 1418.76 | 83.14    | 1434.09|
| Small-10 | 2228.67 | 2228.73 | 2228.67 | 81.27    | 2228.67 | 98.49    | 2245.61|
| Small-15 | 2493.47 | 2493.47 | 2493.47 | 118.41   | 2493.47 | 179.31   | 2555.22|
| Small-20 | 3053.02 | 3053.56 | 3121.43 | 160.38   | 3053.02 | 221.17   | 3176.92|
| Small-25 | 3451.15 | 3451.15 | 3451.15 | 220.20   | 3451.15 | 238.53   | 3552.08|
| Small-30 | 3643.22 | 3643.99 | 3643.22 | 232.84   | 3643.22 | 253.87   | 3774.21|
| Small-35 | 3846.87 | 3848.46 | 3958.73 | 248.13   | 3848.46 | 281.28   | 4022.05|
| Small-40 | 4125.70 | 4128.51 | 4150.79 | 275.58   | 4136.57 | 296.43   | 4394.94|
| Small-45 | 4270.61 | 4276.89 | 4279.19 | 293.91   | 4279.19 | 311.27   | 4594.91|
| Small-50 | 4810.87 | 4831.97 | 4887.16 | 364.84   | 4811.92 | 409.31   | 5090.68|
| T = 6    |         |         |         |          |         |          |        |
| Small-5  | 3299.98 | 3299.98 | 3299.98 | 213.77   | 3299.98 | 249.23   | 3348.43|
| Small-10 | 4832.89 | 4832.89 | 4832.89 | 259.19   | 4832.89 | 301.87   | 4899.86|
| Small-15 | 5566.39 | 5566.39 | 5638.59 | 297.43   | 5566.39 | 366.30   | 5803.08|
| Small-20 | 6833.29 | 6838.42 | 6838.42 | 427.23   | 6833.29 | 528.54   | 7035.02|
| Small-25 | 7454.15 | 7471.42 | 7475.88 | 483.59   | 7475.88 | 631.86   | 7913.47|
| Small-30 | 7847.39 | 7892.29 | 7899.12 | 713.74   | 7868.36 | 843.39   | 8214.21|

MA was able to meet and exceed GA’s results by accurately solving five instance sets with three delivery periods and four instance sets with six delivery periods. These results can be explained by the ability of VNS to intensify searches in each space explored by GA, and find its local minimum. This process allows us to accumulate the local minimums of each part of the space, one of which can be the global minimum that will be the solution of the considered instance. In this case, MA was able to improve the global results obtained by GA by 26.81% and 18.01% respectively for the set of three-period and six-period instances.

We will now analyze the performance of MA compared to HAIR. For the small benchmark consisting of three delivery periods, HAIR provided an optimal solution on three sets of instances and performed better than MA on two sets of instances, while MA found exact solutions for five sets of instances and was able to improve the results obtained by HAIR on two sets of instances. In general, the rate of improvement of HAIR’s results by MA is equal to 1.09% on all instances of three periods. Moving to the small benchmark consisting of six delivery periods, HAIR was able to find exact solutions on three sets of instances while MA outperformed it with an exact resolution of four sets of instances. Each meta-heuristic found a better result for one of the remaining instance sets. As a result, MA still came out on top and gave an overall improvement of 4.1% over HAIR in all six period instance sets.

The improvement in HAIR results was due to the ability of the genetic operators, included in MA, to cover a wider search space and not perform concentrated searches in a single area. In some sets of instances, MA results were biased by search spaces.
with good quality local minimums, so the improvement in results by VNS was not too significant. Moreover, the result retained for each set of instances is the average of a simulation set, not the best result obtained for each set. However, the results obtained by MA did not prevent it from being more competitive than HAIR and meeting the efficiency standards since they are in the range of acceptable solutions.

The error rate of GA and MA compared to the optimal solutions found by the exact method of Archetti et al. (2007) are 29.03% and 2.22% for the three delivery period instances and 25.13% and 7.11% for the six delivery period instances respectively. These results show the effectiveness of hybridizing the GA by adding the VNS local search operator, which significantly improved the GA results, and allowed the accuracy to be approached with a very fine margin of error. These results led to the conclusion that the MA is very competitive.

It is difficult, if not impossible, to make a meaningful comparison based on the execution time required to solve each instance by our methods and those in the literature. This difficulty is reflected in the different implementation environments and in the hardware used by each author. Only a comparison between our two developed meta-heuristics will be established. By comparing the execution time of the two algorithms, we observe that the GA consumes less time to complete the execution process than the MA, on all small instances. This result is expected because the VNS operator is a non-random exploration algorithm, it proceeds iteratively to improve the input result. In fact, the most important component of MA is the VNS, which consumes more than 70% of the total execution time while being essential for the quality of the solution. The VNS can no longer improve the solution from one generation to the next. However, solutions not improved by this procedure can be inserted into the population, thus contributing to better diversity.

The results in this table do not allow us to know which of the two algorithms was able to obtain the optimal solution first. We will only say here that both meta-heuristics are able to obtain good results that support their use in larger benchmarks. We continue our tests by moving now to the large benchmark proposed by Archetti et al. (2012). Table 2 shows the average results obtained by the different meta-heuristics: HAIR, GA-OX and MA, applied on instances ranging from 50 to 200 customers over a planning horizon of six periods. The first column of the Table 2 shows the size of each set of instances. The second, third, and fifth columns give the results of applying HAIR, GA-OX, and MA on the instances of Archetti et al. (2012), respectively. The fourth and sixth columns represent the computational time required to complete the initially set numbers of iterations for GA-OX and MA.

Table 2 shows the optimal solutions of the three meta-heuristics HAIR, GA-OX and MA tested on the large instances proposed by Archetti et al. (2012). In this phase

| Instance | HAIR     | GA-OX     | Times (s) | MA       | Times (s) |
|----------|----------|-----------|-----------|----------|-----------|
| Large-50 | 10,513.07| 10,494.89 | 2460      | 10,494.89| 2520      |
| Large-100| 15,613.61| 15,291.50 | 5460      | 15,291.50| 5580      |
| Large-200| 24,787.14| 24,340.32 | 30,552    | 24,146.13| 35,962    |
of study, we will not focus on the computation time parameter to identify the best meta-heuristic among the three above mentioned ones since the platforms used in our experiments are not the same as those of Archetti et al. (2012). We will only compare the solutions obtained by the different meta-heuristics to identify the most efficient method in terms of solution.

From the Table 2, we can see that MA and GA-OX are better than HAIR and provided new solutions for this benchmark. In fact, the performance of population-based algorithms increases by increasing the variables and parameters of the search space. Indeed, when the number of customers reaches 50 and 100 for a planning horizon that spans six periods, GA-OX and MA improved HAIR’s score by 0.17% for 50-customer instances and by 2.06% for 100-customer instances.

However, both algorithms (GA-OX and MA) were able to obtain the same optimal solution at the end of the experiment for the first two sets of 50 and 100 customer instances. On the other hand, the performance of MA on the 200-customer instances outperformed HAIR and GA-OX. On this set of instances, MA is better than HAIR by 2.58% and GA-OX by 0.80%. This leads us to say that the intensification mechanism linked to the VNS heuristic implemented in MA allowed us to avoid two local minimums that represented the solutions of the other algorithms to obtain a new solution that became the best solution for the instances composed of 200 customers. This allows us to say that MA was able to save on average 1.92% of the total cost of the replenishment network on this set of instances.

As in the case of the small benchmark, a comparison of the execution time between our algorithms and Archetti et al. (2012)’s heuristic is not established because of the difference in the computational systems used by the authors. For this reason, the comparison we will make will only concern the execution time of GA-OX and MA.

Based on the results obtained in this table, we notice that GA-OX is better than MA in terms of computation time for the first two sets of instances because it obtains the same solutions in less time, while MA is better than GA-OX in terms of result and computation time only for the set of instances of 200 customers.

But the computation time mentioned in the table does not indicate the necessary time emitted by the algorithm to obtain the result but rather the execution time needed to complete the number of iterations previously set by the researchers. Therefore, we cannot take these measures as an indicator of performance, and we cannot consider the simulation results obtained in this table.

This leads us to propose to examine the computation time parameter in more detail.

As a result, we will propose a comparison between the two algorithms (MA and GA-OX) based on the time to obtain the optimal solution for each set of instances. A chronological study based on the evolution of the results and the passage from one local minimum to another during the experiments is thus proposed in order to observe how the algorithms progress over the iterations to find their optimal solutions.

From the Figs. 7, 8 and 9, we will identify which algorithm finds the results in less time for each set of large instances. The orange and blue bars in the different figures refer to the computational time taken respectively by GA-OX and MA to obtain the optimal solutions for each set of large instances.

The diagram in Fig. 7 shows the evolution of the results of applying the GA-OX and MA algorithms on large instances consisting of 50 customers. This diagram allows us
to identify the passage of results from one local minimum to another until reaching an optimal solution generated by each algorithm. In this case, the optimal solution is equal to 10494.89 which was reached by MA at time 1860 (s) and by GA-OX at time 2280 (s) i.e. at an interval of 420 seconds which presents a feat for MA which was faster to converge to the optimal solution. We can also notice that in this time interval, GA-OX went through two local minimums before finding the optimal solution.

We will now analyse in detail the time of the GA-OX and MA experiments for the set of 100-customer instances. From the diagram visualized in Fig. 8, we can conclude that MA reached the optimal solution in a very competitive computation time with a deviation from GA-OX of 732 seconds. We can also observe how MA avoided a lot of local minimums and time loss in the exploration phase of the search space, which was noticed with the GA we used.

Finally, we will analyze the evolution of the computation time results of MA and GA-OX on the set of 200-customer instances from the diagram of Fig. 9. With the same model parameters and on the same test instances, MA takes less time than GA-OX to
find the optimal solutions for all instances of Archetti et al. (2007). The observation we made for the set of 100-customer instances was reinforced by the test results for this set of instances, as MA was also able to bypass several local minimums and find the optimal solution provided by GA-OX in a record time of less than 5,712 seconds and furthermore converged to a new optimal solution in less than 14 minutes. Hence, we conclude that MA outperforms GA-OX.

Comparing the results in Table 2 with the results found in Figures 7, 8 and 9, we can conclude that the excessive time taken by MA for each iteration is due to the new structured local search technique that is added to GA-OX. Indeed, the new VNS operator brought a significant improvement in the quality of the obtained solutions and the average execution time to obtain these solutions. The use of VNS allowed a non-random result at each iteration, which explains the fact that the hybrid GA needed more time to restructure the routes and improve the result given by the crossover and mutation operators, and to complete all its iterations. On the other hand, this local search process resulted in better solutions to the problem more quickly.

Moreover, the study of the time needed to obtain optimal solutions allowed us to know that the stopping criterion based on the predefined number of iterations is not the best condition to use in the case of population-based method. This stopping criterion leads to biased results in terms of computation time which must be checked to determine the exact moment of obtaining the optimal solution. A time series analysis of the evolution of the results was necessary in this case.

From the results of our experimentation on the two benchmarks of the Archetti et al. (2007) and Archetti et al. (2012), we can conclude that MA provides competitive solutions for small instances compared to other approximate methods in the literature. Moreover, our results show that MA is the best algorithm used so far to solve large instances of Archetti et al. (2007). We also show the advantage of hybridizing GA with VNS. Finally, GA-OX hybridization has improved the performance of classical GA-OX to better solve large instances and approach exact solutions in very competitive computation times.
5 Conclusion

In this article, we studied a multi-period IRP with deterministic customer demand that is realized over a finite planning horizon. Only one vehicle located at the supplier is used per period to replenish the customers assigned to each delivery route. The objective of our work is to find the best distribution of customers to be served over the delivery periods while avoiding any possibility of stock-out, and to find the route that minimizes the cost of replenishing customers at the supplier.

Our system was solved with two proposed meta-heuristics, namely GA-OX and MA. The development of MA was based on the hybridization of GA-OX with a local search method (VNS). The latter proved its ability to efficiently solve small and large instances of Archetti et al. (2007) and Archetti et al. (2012) and to improve the results obtained by GA-OX in competitive computation times. Moreover, the new hybrid meta-heuristic was able to better explore the search spaces to obtain new and better solutions for large reference instances.

In future works, we will try to improve our MA algorithm by finding other techniques for the calibration of our parameters in order to avoid waste in the execution time while having satisfactory results. Moreover, we plan to use other methods to solve large instances of the IRP problem, such as the savings algorithm or the Lin–Kernighan–Helsgaun heuristic, which have proven to be very practical in solving problems similar to ours, such as in Chen et al. (2019) and Tinós et al. (2018).

We also plan to propose and solve new IRP variants and deal with IRPs involving other inventory management policies or inventory transfer options such as lateral transshipment or making the demand stochastic to approximate reality. The addition of these elements can help explore the benefits of such integration in a supply network and make new contributions to research in this area. Finally, we hope to be able to test our meta-heuristics on real cases in order to really take advantage of the benefits of these methods.

References

Abid, H., Yousaf-Shad, M., Nauman, S., Ijaz, H., Alaa, M., Showkat, G.: Genetic algorithm for traveling salesman problem with modified cycle crossover operator. Comput. Intell. Neurosci. (2017). https://doi.org/10.1155/2017/7430125

Amri Sakhri, M.S.: Comparative analysis of different crossover structures for solving a periodic inventory routing problem. Int. J. Data Sci. Anal. (2021). https://doi.org/10.1007/s41060-021-00280-2

Amri Sakhri, M.S., Tlili, M., Allaoui, H., Korbaa, O.: Order crossover for the inventory routing problem. In: European Symposium on Artificial Neural Networks, pp. 697–702. Bruges, Belgium (2018)

Archetti, C., Bertazzi, L., Laporte, G., Speranza, M.G.: A branch-and-cut algorithm for a vendor-managed inventory-routing problem. Transp. Sci. 41, 382–391 (2007). https://doi.org/10.1287/trsc.1060.0188

Archetti, C., Bertazzi, L., Hertz, A., Speranza, M.G.: A hybrid heuristic for an inventory routing problem. INFORMS J. Comput. 24, 101–116 (2012). https://doi.org/10.1287/ijoc.1100.0439

Azadeh, A., Elahi, S., Farahani, M.H., Nasirian, B.: A genetic algorithm–Taguchi based approach to inventory routing problem of a single perishable product with transshipment. Comput. Ind. Eng. 104, 124–133 (2017). https://doi.org/10.1016/j.cie.2016.12.019

Baniamerian, A., Bashiri, M., Tavakkoli-Moghaddam, R.: Modified variable neighborhood search and genetic algorithm for profitable heterogeneous vehicle routing problem with cross-docking. Appl. Soft Comput. 75, 441–460 (2019). https://doi.org/10.1016/j.asoc.2018.11.029
Bell, W.J., Dalberto, L.M., Fisher, M.L., Greeneld, A.J., Jaikumar, R., Kedia, P., Mack, R.G., Prutzman, P.J.: Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. Interfaces 13(6), 4–23 (1983). https://doi.org/10.1287/inte.13.6.4

Bertazzi, L., Speranza, M.G.: Continuous and discrete shipping strategies for the single link problem. Transp. Sci. 36, 314–325 (2002). https://doi.org/10.1287/trsc.36.3.314.7828

Bertazzi, L., Paletta, G., Speranza, M.G.: Deterministic order-up-to level policies in an inventory routing problem. Transp. Sci. 36, 119–132 (2002). https://doi.org/10.1287/trsc.36.1.119.573

Chen, R.M., Shen, Y.M., Hong, W.Z.: Neural-like encoding particle swarm optimization for periodic vehicle routing problems. Expert Syst. Appl. 138, 112833 (2019). https://doi.org/10.1016/j.eswa.2019.112833

Coelho, L.C., Laporte, G.: An optimized target level inventory replenishment policy for vendor managed inventory systems. Int. J. Prod. Res. 53, 3651–3660 (2015). https://doi.org/10.1080/00207543.2014.986299

Cooray, P.L.N.U., Thashika, D.R.: Machine learning-based parameter tuned genetic algorithm for energy minimizing vehicle routing problem. J. Ind. Eng. (2017). https://doi.org/10.1155/2017/3019523

Ece, Y., Saadettin, E.K.: Multi-trip heterogeneous vehicle routing problem coordinated with production scheduling: memetic algorithm and simulated annealing approaches. Comput. Ind. Eng. 161, 107649 (2021). https://doi.org/10.1016/j.cie.2021.107649

Eleftherios, M., Panagiotis, R., Emmanouil, Z., Christos, T.: Improved branch-and-cut for the Inventory Routing Problem based on a two-commodity flow formulation. Eur. J. Oper. Res. 290(3), 870–885 (2021). https://doi.org/10.1016/j.ejor.2020.08.047

Fardi, K., Ghoushchi, S.J., Hafezalkotob, A.: An extended robust approach for a cooperative inventory routing problem. Expert Syst. Appl. 116, 310–327 (2019). https://doi.org/10.1016/j.eswa.2018.09.002

Gizem, Ö.T., Esra, K., Hande, Y.: An exact solution approach for the inventory routing problem with time windows. Comput. Oper. Res. 134, 105371 (2021). https://doi.org/10.1016/j.cor.2021.105371

Gruler, A., Panadero, J., Armas, J., Moreno-Pérez, J.A., Juan, A.A.: Combining variable neighborhood search with simulation for the inventory routing problem with stochastic demands and stock-outs. Comput. Ind. Eng. 123, 278–288 (2018). https://doi.org/10.1016/j.cie.2018.06.036

Hewitt, M., Nemhauser, G., Savelsbergh, M., Song, J.H.: A branch-and-price guided search approach to maritime inventory routing. Comput. Oper. Res. 40, 1410–1419 (2013). https://doi.org/10.1016/j.cor.2012.09.010

Jahangir, H., Mohammadi, M., Pasandideh Reza, S.H., Neda, Z.N.: Comparing performance of genetic and discrete invasive weed optimization algorithms for solving the inventory routing problem with an incremental delivery. J. Intell. Manuf. 30, 2327–2353 (2019). https://doi.org/10.1007/s10845-018-1393-z

Labadi, N., Prins, C., Reghiliou, M.: A memetic algorithm for the vehicle routing problem with time windows. RAIRO Oper. Res. 42(3), 415–431 (2008). https://doi.org/10.1051/ro:2008021

Mahdi, A., Erfan, B.T., Zahra, K.D., Seyed, R.H., Weiping, D.: An augmented Tabu search algorithm for the green inventory-routing problem with time windows. Swarm Evolut. Comput. 60, 100802 (2020). https://doi.org/10.1016/j.swevo.2020.100802

Meysam, M., Seyed, S.F., Soodabeh, M., Leyla, S.T., Mirpoura, M.: A modified adaptive genetic algorithm for multi-product multi-period inventory routing problem. Sustain. Oper. Comput. 3, 1–9 (2021). https://doi.org/10.1016/j.susoc.2021.08.002

Mirzaei, S., Seifi, A.: Genetic algorithm for traveling salesman problem with modified cycle crossover operator. Comput. Ind. Eng. 87, 213–227 (2015). https://doi.org/10.1016/j.cie.2015.05.010

Mjirda, A., Jarboui, B., Macedo, R., Hanafi, S., Mladenovic, N.: A two phase variable neighborhood search for the multi-product inventory routing problem. Comput. Oper. Res. 52, 291–299 (2014). https://doi.org/10.1016/j.cor.2013.06.006

Mladenovic, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. 24(11), 1097–1100 (1997). https://doi.org/10.1016/S0305-0548(97)00031-2

Molina, J.C., Salmeron, J.L., Eguia, I.: An ACS-based memetic algorithm for the heterogeneous vehicle routing problem with time windows. Expert Syst. Appl. 157, 113379 (2020). https://doi.org/10.1016/j.eswa.2020.113379

Olgun, M.O., Aydemir, E.: A new cooperative depot sharing approach for inventory routing problem. Ann. Oper. Res. 307, 417–441 (2021). https://doi.org/10.1007/s10479-021-04122-z

Park, Y., Yoo, J., Park, H.: A genetic algorithm for the vendor-managed in ventory routing problem with lost sales. Expert Syst. Appl. 53, 149–159 (2016). https://doi.org/10.1016/j.eswa.2016.01.041
Prasetyo, H., Putri, A.L., Fauza, G.: Biased random key genetic algorithm design with multiple populations to solve capacitated vehicle routing problem with time windows. In: Proceedings AIP Conference (2018). https://doi.org/10.1063/1.5042908

Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Comput. Oper. Res. 31(12), 1985–2002 (2004). https://doi.org/10.1016/S0305-0548(03)00158-8

Qin, L., Miao, L., Ruan, Q., Zhang, Y.: A local search method for periodic inventory routing problem. Expert Syst. Appl. 41, 765–778 (2014). https://doi.org/10.1016/j.eswa.2013.07.100

Rau, H., Budiman, S.D., Widyadana, G.A.: Optimization of the multi-objective green cyclical inventory routing problem using discrete multi-swarm PSO method. Transp. Res. Part E Logist. Transp. Rev. 120, 51–75 (2018). https://doi.org/10.1016/j.tre.2018.10.006

Sabar, N.R., Bhaskar, A., Chung, E., Turky, A., Song, A.: An adaptive memetic approach for heterogeneous vehicle routing problems with two-dimensional loading constraints. Swarm Evolut Comput 58, 100730 (2020). https://doi.org/10.1016/j.swevo.2020.100730

Sbai, I., Krichen, S., Limam, O.: Two meta-heuristics for solving the capacitated vehicle routing problem: the case of the Tunisian Post Office. Oper. Res. (2020). https://doi.org/10.1007/s12351-019-00543-8

Tinós R, Helsgaun K, Whitley D (2018) Efficient recombination in the Lin–Kernighan–Helsgaun traveling salesman heuristic. In: International Conference on Parallel Problem Solving from Nature, pp. 95–107. Springer, Cham. https://doi.org/10.1007/978-3-319-99253-2_8

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.