A Memetic Algorithm for the Inventory Routing Problem

Mohamed Salim Amri Sakhri (amri.salim.2013@gmail.com)
Universite de Tunis Institut Superieur de Gestion de Tunis https://orcid.org/0000-0002-7814-6443
Mounira Tlili
Universite de Sousse Institut Superieur du Transport et de la Logistique de Sousse
Ouajdi Korbaa
Universite de Sousse Institut Superieur d'Informatique et des Technologies de Communication de Hammam Sousse

Research Article

Keywords: Inventory Routing Problem , Genetic Algorithm , Memetic Algorithm , Variable Neighborhood Search , Order Crossover , Optimization

Posted Date: September 8th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-848529/v1

Version of Record: A version of this preprint was published at Journal of Heuristics on April 22nd, 2022.
See the published version at https://doi.org/10.1007/s10732-022-09497-1.

Noname manuscript No.
(will be inserted by the editor)

A Memetic Algorithm for the Inventory Routing Problem

Mohamed Salim Amri Sakhri
Mounira Tlili
Ouajdi Korbaa

Received: date / Accepted: date

Abstract In this paper, we investigated an Inventory Routing Problem (IRP) with deterministic customer demand, in a two-tiered supply chain. The supply chain network consists of a supplier who uses a single vehicle with a given capacity to deliver a single type of product to many customers. We are interested in population-based algorithms to solve our problem. A Memetic Algorithm (MA) is developed based on Genetic Algorithm (GA) and Variable Neighborhood Search methods (VNS). The proposed metaheuristics are tested on small and large sizes referenced benchmarks. The results of MA are compared with those of classical GA and with the optimal solutions from the literature. The comparison showed the efficiency of the MA use and its ability to generate high quality solutions within a reasonable time.

Keywords Inventory Routing Problem · Genetic Algorithm · Memetic Algorithm · Variable Neighborhood Search · Order Crossover · Optimization

1 Introduction

Transportation activities are an important part of logistics that is constantly evolving in response to globalization of trade, computerization of processes,

Universite de Tunis, Institut Superieur de Gestion, SMART Research Laboratory, LR11ES03, 2000, Bardo, Tunisia;
E-mail: Mohamedsalim.Amrisakhri@ISG.rnu.tn

Universite de Sousse, Institut Superieur de Transport et de la Logistique, 4023, Sousse, Tunisia;
E-mail: Mounira.Tlili@gmail.com

Universite de Sousse, ISITCom, MARS Research Laboratory, LR17ES05, 4011, Hammam Sousse, Tunisia;
E-mail: Ouajdi.Korbaa@MARS.rnu.tn

new consumer patterns and environmental requirements. Logistics managers coordinate the entire supply chain, including the various order picking operations, procurement, fleet management, transport planning, warehousing, and inventory management.

In this field, information and communication technologies are becoming increasingly important. They ensure the integrated management between the various compartments of the distribution network, the geolocation, the product traceability, and inventory management. In addition, integrated management has created new challenges to be considered. New computer skills should solve these different types of problems related to inventory, business relationships and eco-driving (less fuel consumption, minimization of access time to stock and minimization of inventory costs).

The Inventory Routing Problem (IRP) is a distribution problem combining two activities of the supply chain: inventory management and routing of vehicles. It aims to determine, in each period, which customer could be visited, the amount of product to be delivered and the route of the vehicle. Solving this problem comes down to minimizing the inventory holding costs of the customers and the supplier 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., [1]. They considered IRP as a process of distributing a single product over several time periods, from one supplier to a set of customers, following constant (deterministic) demand over time, with an infinite and homogeneous fleet of vehicles. Since it appeared, many variations of IRP have been proposed in the literature.

In this paper, we are interested in research which considered a periodic IRP with a deterministic request, as in the work of [2]. The IRP variants known in this research axis have dealt with the single-product case as in [4] or the multi-product case studied in [5]. Most researchers have studied IRPs with a vehicle fleet made up of a single vehicle or of several vehicles, respectively by [4,5]. In the case of the multi-vehicle IRP, the types of vehicle fleet can be considered are either homogeneous as in [3] or heterogeneous as in [6]. 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 [14,9,8].

IRP solving methods are mainly classified into two categories which are exact and approximate methods. An exact method is a resolution method that guarantees obtaining an optimal solution to a given optimization problem. Linked to IRP resolution, the most well-known exact methods are branch-and-cut, branch-and-price and branch-price-and-cut used respectively by [11–13]. The approximate methods consist in finding a satisfactory solution to an optimization problem at a reasonable computing time, especially when the problem search area is large and complex. These kind of methods have been applied successfully in the resolution of the different IRP variants and in particular the metaheuristics which are the approaches most used in this line of work. The most used metaheuristics in solving IRP variants are Variable Neighbor-

hood Search, Tabu Search, Genetic Algorithm, Ant Colony Algorithm, Particle Swarm Optimization applied respectively by [18–22].

In this paper, we consider one of the most basic and well-studied IRP under the Vendor Managed Inventory (VMI) system. This problem is introduced and solved with an exact method in [9]. Effective approaches have been proposed to solve the same model, as the metaheuristic of [7] called Hybrid Approach to Inventory Routing (HAIR) combined a tabu search scheme with ad hoc designed mixed integer programming models. Here, we will propose an efficient metaheuristic to solve the aforementioned problem called Memetic Algorithm (MA). MA has often been used to solve Vehicle Routing Problem (VRP) and it has been shown to be effective in producing high quality results, [23–25]. However, MA has not yet been studied for IRP, which prompted us to do so.

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

- Development of a Genetic Algorithm 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 Variable Neighborhood Search 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 remainder of the paper is organized as follows. In the next section, we formally describe our problem, and present a mathematical programming formulation for the IRP. The third section introduces the different algorithms developed to solve the proposed problem. In the fourth section, we present the results of in-depth computational experiments, analyze the relationships between inventory and transport costs, and compare our solutions to those in the literature. In the last section, we conclude our work and give some perspectives for future research.

2 IRP Model

In this paper, we are interested in the study, analysis and resolution of an NP-Hard problem, which is IRP. Our goal is to optimize the total supply chain costs and avoid stockouts for customers. In this section, we will describe the

mathematical model to understand the resolution structure of our problem. The considered IRP can deal with four issues: the management of customer and supplier stocks, the quantities of stock to be delivered to avoid stock shortages, assignment of customers to delivery period, design, and optimization of the vehicle routes.

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 $N = \{0, ..., n\}$, 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_i^t is consumed by the customer $i \in N'$.

The binary variable Y_{ij}^t is equal to 1 if the edge $[i, j]$ is traversed by the vehicle at period $t \in T$ with $(i, j) \in N$, otherwise it is equal to 0. The binary variable x_i^t 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_i^t 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_i^0. 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_i^t is the quantity delivered to a customer i at period $t \in T$. The variable v_i^t 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 [9]. The objective function is to minimize the total operational costs, formulated as follows.

$$\min \sum_{t}^{T} h_0 I_0^t + \sum_{t}^{T} \sum_{i}^{N'} h_i I_i^t + \sum_{t}^{T} \sum_{i}^{N} \sum_{j}^{N} C_{ij} Y_{ij}^t \tag{1}$$

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 talk about constraints that have the same functionality in a separate subsection.

2.3.1 Inventory Level Constraints

The constraint of inventory level at the supplier at the period t:

$$I_0^t = I_0^{t-1} + p^t - \sum_{i}^{N'} q_i^t \qquad \forall\, t \in T \tag{2}$$

The constraint of inventory level at the customers at the period t:

$$I_i^t = I_i^{t-1} - r_i^t + q_i^t \qquad \forall\, i \in N',\ \forall\, t \in T \tag{3}$$

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

$$I_i^t \geq L_i \qquad \forall\, i \in N',\ \forall\, t \in T \tag{4}$$

$$U_i \geq I_i^t \qquad \forall\, i \in N',\ \forall\, t \in T \tag{5}$$

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:

$$q_i^t \geq U_i x_i^t - I_i^{t-1} \qquad \forall\, i \in N',\ \forall\, t \in T \tag{6}$$

$$q_i^t \leq U_i - I_i^{t-1} \qquad \forall\, i \in N',\ \forall\, t \in T \tag{7}$$

$$q_i^t \leq U_i x_i^t \qquad \forall\, i \in N',\ \forall\, t \in T \tag{8}$$

2.3.3 Vehicle Routing Constraints

The flow conservation constraint:

$$\sum_{i}^{N} Y_{ij}^{t} = \sum_{i}^{N} Y_{ji}^{t} \qquad \forall\, j \in N',\ i \neq j,\ \forall\, t \in T \tag{9}$$

The vehicle capacity constraint:

$$\sum_{i}^{N'} q_{i}^{t} \leq Q x_{i}^{t} \qquad \forall\, t \in T \tag{10}$$

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

$$v_{j}^{t} - v_{i}^{t} + Q Y_{ij}^{t} \leq Q - q_{i}^{t} \qquad \forall\, i, j \in N',\ i \neq j,\ \forall\, t \in T \tag{11}$$

$$q_{i}^{t} \leq v_{i}^{t} \leq Q \qquad \forall\, i \in N',\ \forall\, t \in T \tag{12}$$

2.3.4 Non-negativity and Integrality Constraints

$$Y_{ij}^{t} \in \{0, 1\} \qquad \forall\, i, j \in N',\ i \neq j,\ \forall\, t \in T \tag{13}$$

$$x_{i}^{t} \in \{0, 1\} \qquad \forall\, i \in N',\ \forall\, t \in T \tag{14}$$

$$I_{i}^{t} \geq 0 \qquad \forall\, i \in N,\ \forall\, t \in T \tag{15}$$

$$q_{i}^{t} \geq 0 \qquad \forall\, i \in N',\ \forall\, t \in T \tag{16}$$

$$v_{i}^{t} \geq 0 \qquad \forall\, i \in N',\ \forall\, t \in T \tag{17}$$

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) to (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) to (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) to (12) will be used to define the initial solutions of GA and MA. Inventory level constraints (2) and (3) are updated at the end of each period to prepare for the next replenishment cycle. The constraints (13) to (17) will ensure the feasibility of the model.

3 Metaheuristic Description

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

The metaheuristics developed consist of two phases of optimization. In the first phase, the sorting algorithm tries to find all the customers to serve at each period while respecting the delivery priority constraints. The algorithm also defines the quantities to be delivered while respecting the vehicle capacity constraints, the quantity of stock available at the supplier and the customer's admissible stock level. It ends up generating a first solution (a route) not optimized.

The second phase consists in optimizing the initial solution obtained by means of genetic operators and the local search method described later. In this phase, we will look for the path that minimizes the distance travelled by the vehicle to meet customer demand without violating the vehicle routing constraints. In this paper, two metaheuristics are proposed to solve the IRP model. The first is GA, consisting of the two-point crossover using the Order Crossover technique (OX) and the classic mutation operator which works randomly. The general structure of the GA developed is described in Figure 1.

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. Next, GA proceeds to the exploration of the search space with the use of crossover and mutation operators. The aim of these genetic operators is to enrich the diversity of the population by manipulating the chromosomes structure (routes).

The second metaheuristic is MA. This algorithm retains the same steps of GA by adding a local search operator, which is the VNS after the mutation operator. The VNS operator serves 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 thoroughly explored 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 for each metaheuristic is described in the following subsections.

3.1 General Structure of Genetic Algorithm

In this sub-section, we will describe the functional process of the GA and present its different operators. Then, we will introduce the local search operator (VNS) which will allow to transform our GA into MA.

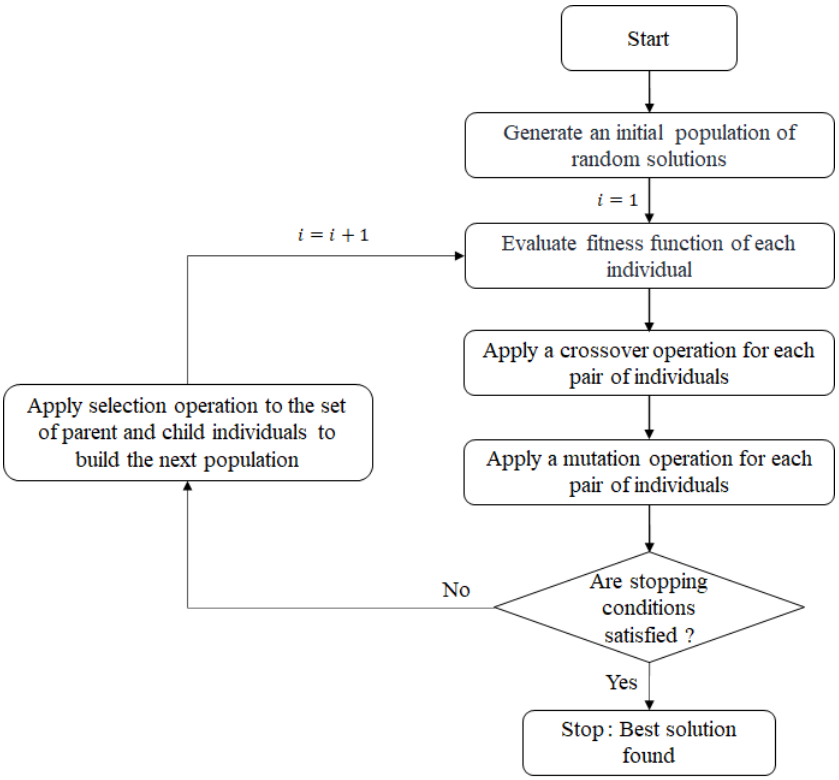

Fig. 1 Flowchart of Genetic Algorithm (GA).

3.1.1 Construction of an initial solution

The main objective of the construction phase is to determine the set of customers who are at risk of being out of stock during each period and those who are minimizing the supply chain costs. This step considers both the inventory level and the storage cost at the customers. The constraints could be taken into account are the inventory level equation and the inequality of the minimum inventory level, and we need also the storage cost parameter of the supplier and customers.

The choice of the initial population is important because it influences the speed of convergence of the algorithms to the global optimum. Construction of the initial solution is a way to control the quantity of stock (transferred and on hand), at the different nodes of the distribution network, to avoid stock-out in each period and minimize total storage costs. Control of the transported stock is due to the constraint of the vehicle capacity and the quantity to be delivered is determined by the order 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 clients 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 to generate an initial population of P chromosomes, each chromosome is made up of two individuals. This population represents the set of delivery routes used as a basis for future generations. The set of individuals is generated randomly, each individual is made up of customers selected by the ranking algorithm that we have just seen in the previous paragraph.

3.1.4 Evaluation function

The function of evaluating an individual is the most sensitive part of our algorithm. This function uses all the components of the model in a simulation that gives a result known as the chromosome fitness value. This result is evaluated during the remainder of the GA steps in order to determine a chromosomal structure having an improved fitness value. The fitness value of chromosomes is determined by the cost of vehicle routes and the inventory holding costs at the different nodes of 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. Suitability for individuals must be determined, as it is necessary for the selection and replacement steps.

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

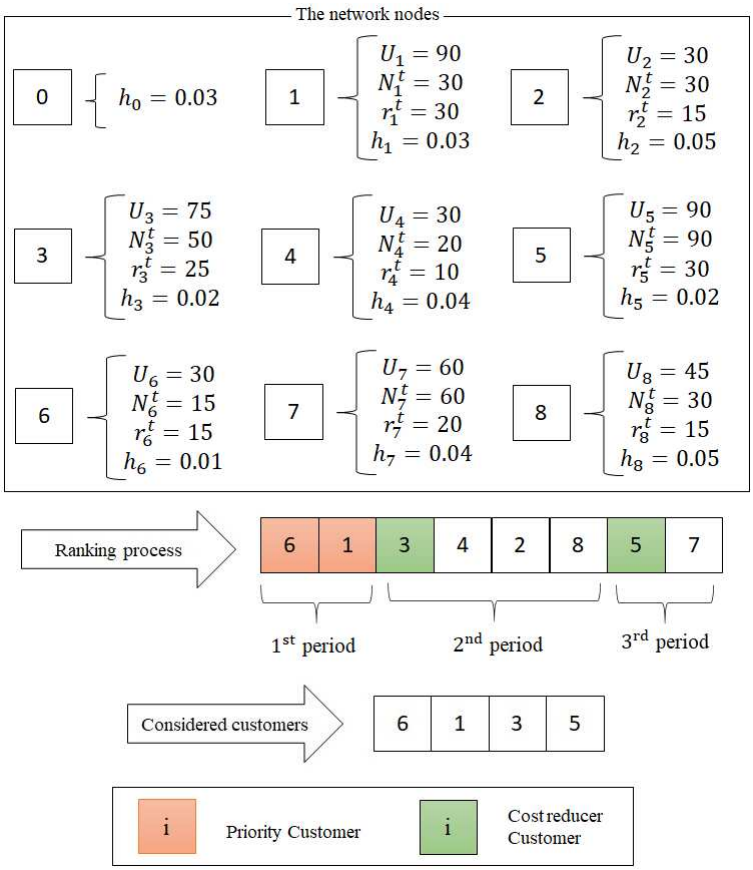

Fig. 2 Ranking process of customers

3.1.5 The selection operator

This operator is used to assess the individuals who are more inclined to obtain the best results. So that all individuals have the same chances of being chosen and of being 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 adaptive value intervention. Therefore, 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 to optimize only the distance travelled by the vehicle without considering the other objectives defined in the previous step. Three genetic

operators will be used here which are the crossover, mutation, and replacement operators. Next, we will transform the GA to MA by adding VNS after the GA mutation operator. 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 [17,16]. This type of crossover takes place as follows.

A two-point crossover is performed between the first and the last gene of the chromosome. The genes are coded with integer numbers. The positions of the two cut points are selected randomly. First, the genes between the cut points are copied in the same order at the same position in the offspring. Then, from the second cut point of one of the parents, the genes of the other parent are copied in the same order, omitting the existing genes.

The example in the Figure 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.

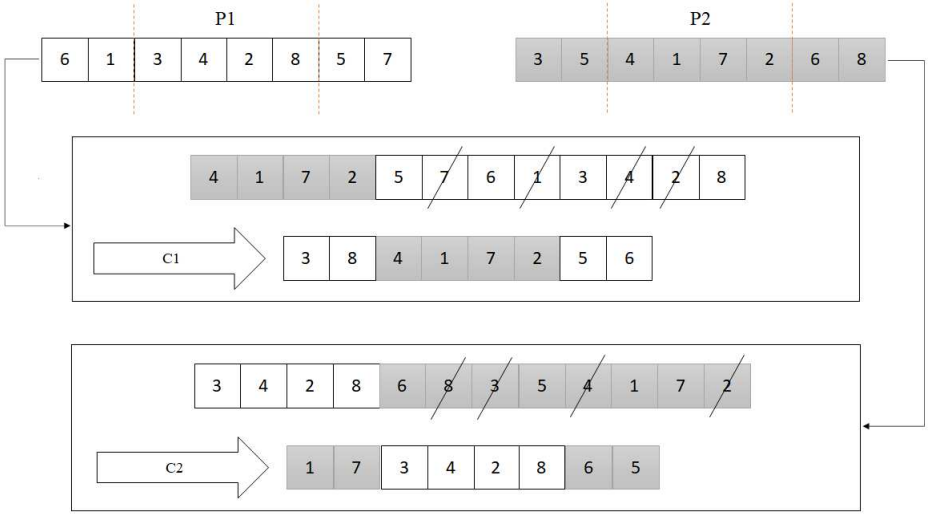

Fig. 3 Order Crossover OX process

3.2.2 Mutation

The mutation is considered to be the operator responsible for maintaining genetic diversity in a population. The mutation operator brings the ergodicity property of space exploration by GA. This property indicates that the GA can reach all points in the search space without iterating through all of them in the resolution process. The convergence properties of GA are, therefore, theoretically strongly dependent on this operator. The performances of the GA will be improved by applying at each iteration a mutation operation at each new generation, in which two genes from the same individual will be selected at random and their values will be 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 following the successive application of the operators of selection, crossing and mutation and to eliminate the bad ones. It allows the individuals of the parent and offspring with the best fitness value to be reintroduced into the new population. Before inserting them into the new population, the individuals must be evaluated according to their fitness to choose the most suitable ones which reduce the total cost of the network.

In our work, we have chosen the tournament technique as the selection operator. This technique uses a random selection of pairs of parents. Then, the latter will be compared to their offspring according to their fitness. Finally, we will choose the couple of individuals with the highest fitness score. The operating mode of this operator is described in the following figure.

In the example of the Figure 4, we select from among the set of chromosome pairs (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 the score of their fitness will allow them to win several times, this favours the survival of their genes.

3.2.4 Stop condition

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

3.3 Memetic Operator

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

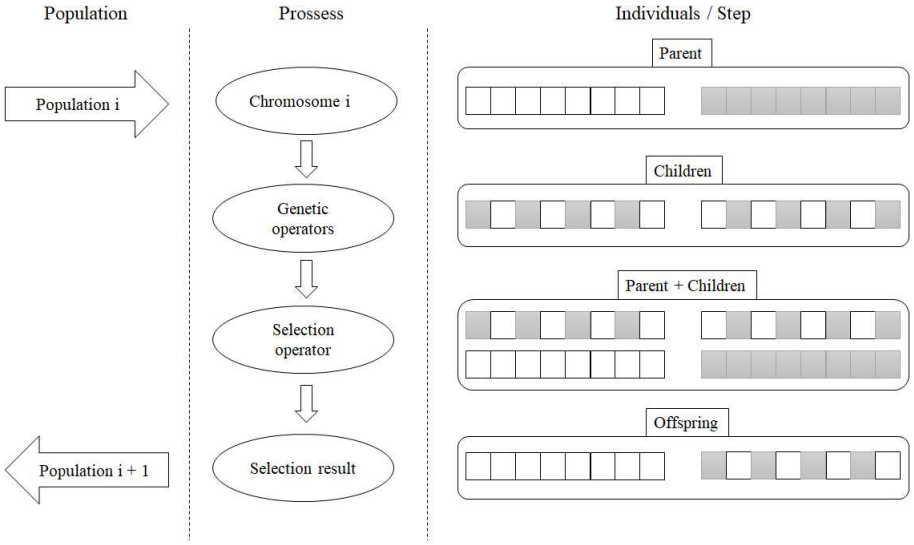

Fig. 4 Selection Process of GA

To avoid this behaviour, there are local search methods that optimize the searches in each space browsed by the GA. To reap the benefits of these methods, we will use a powerful hybrid GA known as Memetic Algorithm (MA). Indeed, only one change in the structure of the GA will be made. A local search method, which is Variable Neighbourhood Search (VNS) will be introduced after the application of the mutation. In this subsection, we will present the hybridization operator VNS that will allow our GA to improve its performance.

3.3.1 Variable Neighbourhood Search (VNS)

The great strength of GA is their ability to cover the area of the solutions space containing the different local optimum 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. It is, therefore, natural to think of associating a local search algorithm with the GA to find the optimum exact values.

This optimization can easily be done by applying a local search method to the best paths found by GA. In fact, the GA-VNS association is a virtual necessity, the two methods are complementary and have different fields of application. GA eliminates the combinatorial aspect of the problem. This allows 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 following the random mutation process presented previously. In this step, our algorithm will opt for changes in the positions of the customers which generates a path at the least

cost. To do this, we will use the VNS method developed in [15]. VNS is based on a process of change of path, favouring a phase of diversification which 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 neighbourhoods. Its basic idea is to perform an interchange in two phases: a local optimum is obtained during the **Local Search Phase** and a perturbation of the solution is carried out during the **Shaking Phase**. We will develop these different phases of the VNS in what follows.

3.3.2 The VNS Strategy for Fitness Improvement

The application of VNS as a local search method in memetic algorithms to solve VRP problems is rare, despite its efficiency which has been proven by some researchers, as in [10]. In this paragraph, we will describe the VNS process applied to a vehicle routing case.

- We consider an initial solution s.
- Shaking: We generate a neighbouring solution $s1 \in M_1(s)$, 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 Figure 5.

The next example of the Figure 6 shows how to create the first child C1* with the VNS process. First, we randomly selected the first node i. Then we change the position of its successor node (next node) with the first node of the neighbourhood list predefined for the selected node. If changing the position of the nodes improves the function of the route cost, then we proceed to this change. Otherwise, we randomly select another node from the list and restart the process. This process is repeated until we find a new node that reduces the total cost of the route. In the example of Figure 6, the code of the randomly selected customer is 4, in C1. If we are to test all customers in the neighbourhood of this customer, the successor customer code that reduces the total cost of the route will be 3, the switch will be done between node 2 and 3, and the new structure of C1 will be 0-1-4-3-2-0.

By adding the local search operator VNS, we aim to improve GA solutions. This will be the challenge of our next calculation tests.

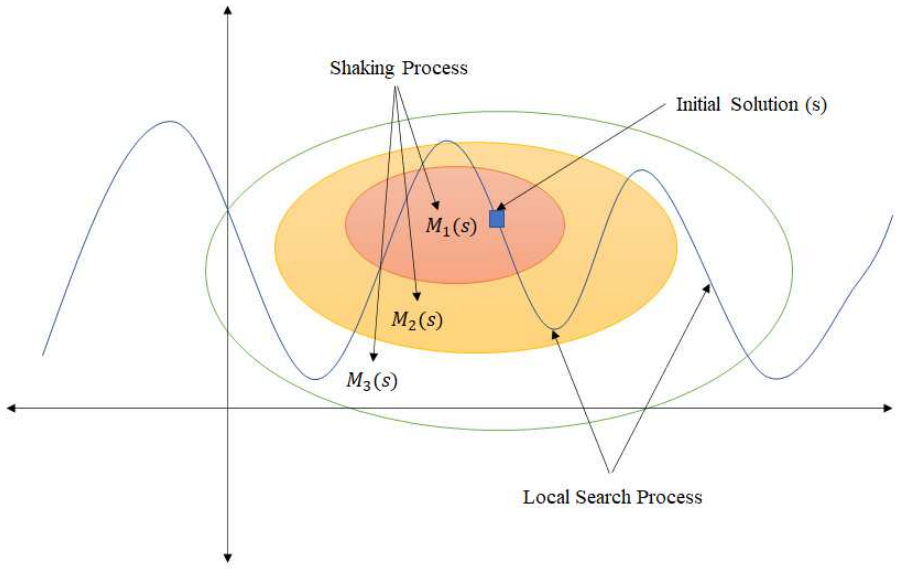

Fig. 5 VNS working process

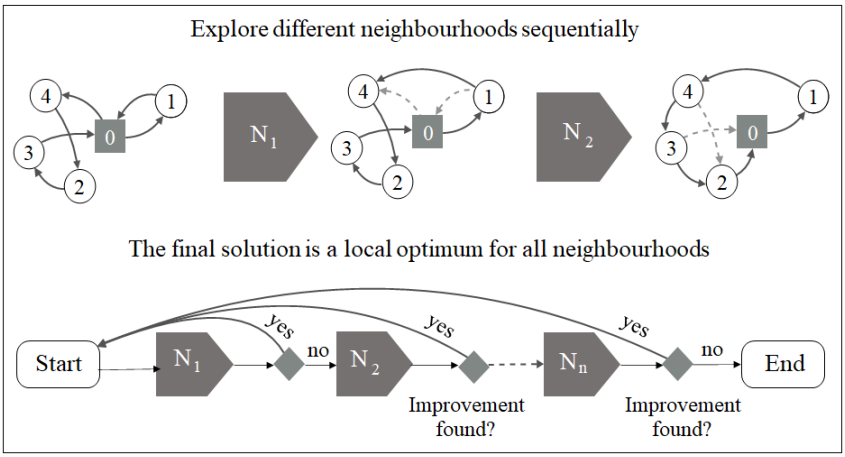

Fig. 6 The process of exchanging customers with VNS

4 Implementation and benchmarks

The developed GA and MA were implemented in Java EE-like for ECLIPSE environment and run on a PC with a 2.40 GHz Intel® Core™ i7-5500U CPU @ processor and 8 GB of RAM. We applied our algorithm to the set of small-size instances proposed in [9], then to the large-size instances given in [7] 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_i^t 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 : $\sum_i^N {}' r_i^t$;
- Maximum inventory level U_i at customer i : it is given by $r_i^t * g_i$ 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 : $\sum_i^N {}' U_i$;
- Starting inventory level I_i^0 at the customer $i : U_i - r_i^t$;
- 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 : $3/2 * \sum_i^N {}' r_i^t$;
- 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 [14].

5 Computational Results

In this sub-section, we test the performance of the developed GA and MA applied on the benchmarks of [9,7]. For each metaheuristic, the crossover probability is $P_c = 0.5$, and the mutation probability is $P_m = 0.1$. For MA, we add the VNS probability, which is $P_{vns} = 0.1$. The initial population, for each heuristic, contain 25 chromosomes (50 pathways), for instances up to 50 customers and contain 50 chromosomes (100 pathways), for instances up to 200 customers. For the first benchmark, the maximum number of iterations is 200 for the small instances up to 20 customers, 400 for the rest of small instances. For second benchmark, the maximum number of iterations is 1000. Note that, each instance of [9] is running 30 times, the average results of these runs are considered the solution of the executed instance.

The chosen problem was initially solved by the BC algorithm of the Cplex solver, in [9]. The mentioned method has shown its limits in terms of result and calculation time by increasing the size of the problem. For instances with three delivery periods, the number of customers considered must 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 computing time by increasing the complexity of

the problem. To face larger problems like that of [7], it is useful to use heuristic methods such as the case of [7], 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 following paragraph.

Table 1 shows the results obtained by our developed metaheuristics and those of the previous investigations, used in the same benchmark of [9]. The first column indicates the size of the instance sets. The second column presents the optimal solution to the problem solved on [9] with the BC 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 of [7], GA-OX and MA developed in this article. 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 [9]. Note that UB is a limit beyond which the result becomes unacceptable.

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

Table 1 Average Result Values obtained by the different Resolution Methods for the Small-Instances

The results of our tests and those of the researchers who considered the small instances of [9]. are displayed in Table 1. As a first observation, we note that the developed algorithms have respected the intervals of the solutions determined by [9] for all the sets of instances. This leads us to dig deeper into our research and see the benefits of using these two algorithms. We will start by comparing the results of the GA to those of the MA. This will allow us to identify which of the two algorithms is the most efficient at solving the considered problem. Next, we will compare the results of our best algorithm with those of HAIR. Finally, we will determine the error rate of these algorithms based on the exact solution to the problem.

GA was able to resolve exactly four of the ten instance sets with a planning horizon consisting of three periods, and two of the six instance sets with a planning horizon equal to six periods. MA was able to achieve and surpass GA's results by accurately solving five instance sets of three-delivery period and four instance sets of six-delivery period. These results can be explained by the ability of the VNS to intensify searches in each space explored by the GA, and to find its local minimum. This process allows us to accumulate local minimums of each part of space, one of which may be the global minimum that will be the solution of the considered instance. In this case, MA was able to improve the overall results obtained by GA with 26.81% and 18.01% respectively for the set of instances of three periods and those of six periods.

We are now going to analyse the performances of MA compared to those of HAIR. For the small benchmark consisting of three delivery periods: HAIR gave 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 improvement rate of HAIR results by MA is equal to 1.09% on all instances of three periods. For the small benchmark consisting of six delivery periods: while HAIR was able to find exact solutions on three sets of instances, MA outperformed it with four exact solutions. Each metaheuristic found a better result for one of the remaining instances sets. Therefore, MA is still in the lead and resulted in an overall improvement of 4.1% over HAIR in all sets of instances of six periods.

The improvement in HAIR results was made due to the ability of the genetic operators, included in MA, to cover a larger search space and not to perform searches focused on a single area. In some sets of instances, MA results was biased by search spaces that show good quality local minimums, so the improvement in results by VNS was not too significant. In addition, the result retained for each set of instances is the average of a simulation set and 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 within 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 [9] are respectively 29.03% and 2.22% for the instances of three delivery periods and 25.13% and 7.11% for the instances of six delivery periods. These results show the efficiency of the GA hybridization by the addition of the local search operator VNS, which significantly improved the GA results, and which made it possible to approach the precision within a very fine margin of error. These results led to the conclusion that MA is very competitive.

It is difficult if not impossible to make a meaningful comparison based on the execution time required to resolve each instance by our methods and those in the literature. This difficulty is reflected in the different implementation environments and in the material used by each of the authors. Only a comparison between our two developed metaheuristics 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 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 run-time while being essential for the quality of the solution. 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, contributing to better diversity.

The results of 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 the two metaheuristics are able to obtain good results which support their use in broader benchmarks. We continue our tests by switching now to the large size benchmark proposed by [7]. Table 2 illustrates the average results obtained by the different metaheuristics: HAIR, GA-OX and MA, applied on instances going from 50 to 200 customers over a planning horizon of six periods. The first column of Table 2 shows the size of each set of instances. The second, third and fifth columns respectively give the results of the application of HAIR, GA-OX and MA on the instances of [7]. The fourth and sixth columns represent the computation time required to complete the iteration numbers initially set for GA-OX and MA.

Instance	HAIR	GA-OX	Times (s)	MA	Times (s)
Large-50	10513,07	10494,89	2460	10494,89	2520
Large-100	15613,61	15291,50	5460	15291,50	5580
Large-200	24787,14	24340,32	30552	24146,13	35962

Table 2 Average Result Values for the Large-Instances by HAIR, GA-OX and MA

Table 2 shows the calculation results of the three different metaheuristics applied to the large size instances proposed by [7]. From Table 2, we can compare the results of the different methods in order to identify the most efficient one in terms of result. From the table, the results of MA and GA-OX are better than those of HAIR and provided new solutions for this benchmark.

When the number of clients equals 50 and 100, GA-OX and MA were able to obtain the same optimal solution and improved the HAIR score by 0.17% for instances of 50 customers and by 2.06% for those of 100 customers. In contrast, MA's performance on instances of 200 customers is better than HAIR by 2.58% and GA-OX by 0.80%. This leads us to say that MA avoided two local minimums to obtain a new solution that has become the best solution for instances of 200 clients. MA allow us to save on average 1.92% of the total cost.

As in the case of the small benchmark, a comparison of the execution time between our algorithms and the heuristic of [7] is not established by considering the difference in the computational systems used by the authors. For this reason, the comparison we are going to make will only affect the run-time of GA-OX and MA. In Table 2, the calculation time mentioned does not

indicate the time necessary to obtain the result, but the time emitted by the algorithm to carry out the number of iterations fixed in advance. Therefore, we cannot take these measurements as an indicator of performance and we will only consider the simulation results obtained in this table and review the calculation time values later.

Table 3 will allow us to identify which developed algorithm applied in the benchmark of [7] finds its results in less time. The first column indicates the size of each set of instances. The second and third columns give respectively the results and the computation time of the application of GA-OX to obtain this result for each set of instances. The fourth and fifth columns represent respectively the results and the computation time of applying MA to obtain this result for each set of instances.

Instance	GA-OX	Times (s)	MA	Times (s)
Large-50	10494,89	2280	10494,89	1860
Large-100	15291,50	5358	15291,50	4626
Large-200	24340,32	30552	24146,13	25638

Table 3 Average Computational Time to obtain Optimal Large-Instances Results by GA-OX and MA

This table not only provides the best results of the two algorithms, but also the associated computation time. This information makes it possible to know when the algorithms were able to improve the results from one iteration to another. Now, it is possible to make a comparison of the execution times necessary for each of the two aforementioned algorithms to acquire the best results.

The performance of the MA is clearly visible in Table 3. With the same model settings and on the same test instances, MA takes less time to find the optimal solution than GA-OX for all instances. The excessive time taken by MA for each iteration is due to the new structured local search technique that is added to the GA. The use of VNS allowed a non-random result in each iteration, for this the hybrid GA needed more time to restructure the routes and improve the result given by the crossover and mutation operators. This made it possible to find better solutions to the problem more quickly.

Indeed, the new operator VNS brought a significant improvement in the quality of the solutions obtained and in the average run-time to obtain its solutions. For instances of 50 customers, MA took less than 7 minutes than GA-OX to find the optimal solution. MA reduced GA-OX compute time by approximately 13 minutes, for instances of 100 customers. For instances of 200 customers, MA took less than 96 minutes to achieve the same result as GA-OX. For the same instances, MA took about 14 minutes longer to get a new optimal solution, which was not achieved by GA-OX, while remaining competitive in terms of compute time.

Moreover, Table 3 allowed us to know that the stop criteria based on the number of the iteration is not the best condition to use in the case of hy-

bridization of a method. This stop criteria allows biased results in terms of computation time that must be checked to determine the exact moment to obtain a solution. A chronological analysis of the evolution of the solutions proved to be more effective in this case.

From the results of the two benchmarks of [9,7], we can conclude that MA allows to have competitive solutions for small size instances compared to other approximate methods in the literature. In addition, our results show that MA is the best algorithm used so far to solve large size instances. We have also shown the advantageous of GA hybridization with VNS. In fact, GA hybridization has improved the performance of classic GA to better resolve large size instances. It also made it possible to get closer to exact solutions in very competitive computing times.

6 Conclusion

In this paper, we investigated a multi-period IRP with deterministic customer demand that is realized over a finite planning horizon. A single vehicle located at the supplier is used per period to replenish the customers assigned to each delivery route. We have solved our IRP with two metaheuristics, which are GA-OX and MA. The development of MA was based on GA-OX hybridization with a local search method VNS. The latter has proven its ability to efficiently solve small and large instances of [9,7] and to improve the results obtained by GA-OX in competitive computing times. In addition, the new hybrid metaheuristic could better explore search spaces to obtain new better-quality solutions for large instances of [7].

In the work to come, we think to develop new hybrid metaheuristics resulting from the genetic algorithm, use them to solve large instances and, compare their results with those of MA. We also plan to implement new variants of IRP as well as other inventory management policies or include other techniques for transferring inventory between customers such as lateral transshipment. These can help explore the benefits of such integration into a distribution network and make new contributions to research in this area. Also, we plan to test our metaheuristics on real cases to really take advantage of the benefits of these methods.

References

1. Bell WJ, Dalberto LM, Fisher ML, Greenfield AJ, Jaikumar R, Kedia P, Mack RG, Prutzman PJ (1983) Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. Interfaces, 13 (6), 4–23. https://doi.org/10.1287/inte.13.6.4
2. Qin L, Miao L, Ruan Q, Zhang Y (2014) A local search method for periodic inventory routing problem. Expert Systems with Applications, 41, 765–778. https://doi.org/10.1016/j.eswa.2013.07.100
3. Coelho LC, Cordeau JF, Laporte G (2012) Consistency in multivehicle inventory routing. Transportation Research Part C: Emerging Technologies, 24, 270–287. https://doi.org/10.1016/j.trc.2012.03.007

4. Bertazzi L, Speranza MG (2002) Continuous and discrete shipping strategies for the single link problem. Transportation Science, 36, 314–325. https://doi.org/10.1287/trsc.36.3.314.7828
5. Mjirda A, Jarboui B, Macedo R, Hanafi S, Mladenovic N (2014) A two phase variable neighborhood search for the multi-product inventory routing problem. Computers & Operational Research, 52, 291–299. https://doi.org/10.1016/j.cor.2013.06.006
6. Persson JA, Gothe-Lundgren M (2005) Shipment planning at oil refineries using column generation and valid inequalities. European Journal of Operational Research, 163, 631—652. https://doi.org/10.1016/j.ejor.2004.02.008
7. Archetti C, Bertazzi L, Hertz A, Speranza MG (2012) A hybrid heuristic for an inventory routing problem. INFORMS Journal on Computing, 24, 101–116. https://doi.org/10.1287/ijoc.1100.0439
8. Coelho LC, Laporte G (2015) An Optimized Target Level Inventory Replenishment Policy for Vendor Managed Inventory Systems. International Journal of Production Research, 53, 3651–3660. https://doi.org/10.1080/00207543.2014.986299
9. Archetti C, Bertazzi L, Laporte G, Speranza MG (2007) A branch-and-cut algorithm for a vendor-managed inventory-routing problem, Transportation Science, 41, 382-–391. https://doi.org/10.1287/trsc.1060.0188
10. Sbai I, Krichen S, Limam O (2020) Two meta-heuristics for solving the capacitated vehicle routing problem: the case of the Tunisian Post Office, Operational Research. https://doi.org/10.1007/s12351-019-00543-8
11. Coelho LC, Laporte G (2013) A branch-and-cut algorithm for the multi-product multivehicle inventory-routing problem. International Journal of Production Research, 51, 7156–7169. https://doi.org/10.1080/00207543.2012.757668
12. Hewitt M, Nemhauser G, Savelsbergh M, Song JH (2013) A Branch-and-Price Guided Search Approach to Maritime Inventory Routing. Computers & Operations Research, 40, 1410–1419. https://doi.org/10.1016/j.cor.2012.09.010
13. Desaulniers G, Rakke JG, Coelho LC (2015) A branch-price-and-cut algorithm for the inventory-routing problem. Transportation Science, 50, 1060–1076. https://doi.org/10.1287/trsc.2015.0635
14. Bertazzi L, Paletta G, Speranza MG (2002) Deterministic Order-Up-To Level Policies in an Inventory Routing Problem. Transportation Science, 36, 119–132. https://doi.org/10.1287/trsc.36.1.119.573
15. Mladenovic N, Hansen P (1997) Variable neighborhood search. Computers & Operations Research, 24 (11), 1097—1100. https://doi.org/10.1016/S0305-0548(97)00031-2
16. Labadi N, Prins C, Reghioui M (2008) A memetic algorithm for the vehicle routing problem with time windows. RAIRO - Operations Research, 42(3), 415–431. https://doi.org/10.1051/ro:2008021
17. Prins C (2004) A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research, 31(12), 1985–2002. https://doi.org/10.1016/S0305-0548(03)00158-8
18. Gruler A, Panadero J, Armas J, Moreno-Pérez JA, Juan AA (2018) Combining variable neighborhood search with simulation for the inventory routing problem with stochastic demands and stock-outs. Computers & Industrial Engineering, 123, 278–288. https://doi.org/10.1016/j.cie.2018.06.036
19. Mirzaei S, Seifi A (2015) Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator. Computers & Industrial Engineering, 87, 213–227. https://doi.org/10.1016/j.cie.2015.05.010
20. Azadeh A, Elahi ,S Farahani MH, Nasirian B (2017) A genetic algorithm-Taguchi based approach to inventory routing problem of a single perishable product with transshipment. Computers & Industrial Engineering, 104, 124–133. https://doi.org/10.1016/j.cie.2016.12.019
21. Fardi K, Ghoushchi SJ, Hafezalkotob A (2019) An extended robust approach for a cooperative inventory routing problem. Expert Systems with Applications, 116, 310–327. https://doi.org/10.1016/j.eswa.2018.09.002
22. Rau H, Budiman SD, Widyadana GA (2018) Optimization of the multi-objective green cyclical inventory routing problem using discrete multi-swarm PSO method. Transportation Research Part E: Logistics and Transportation Review, 120, 51–75. https://doi.org/10.1016/j.tre.2018.10.006

23. Sabar NR, Bhaskar A, Chung E, Turky A, Song A (2020) An Adaptive Memetic Approach for Heterogeneous Vehicle Routing Problems with two-dimensional loading constraints. Swarm and Evolutionary Computation, 58, 100730. https://doi.org/10.1016/j.swevo.2020.100730
24. Molina JC, Salmeron JL, Eguia I (2020) An ACS-based memetic algorithm for the heterogeneous vehicle routing problem with time windows. Expert Systems with Applications, 157, 113379. https://doi.org/10.1016/j.eswa.2020.113379
25. Ngueveu SU, Prins C, Calvo RW (2010) An effective memetic algorithm for the cumulative capacitated vehicle routing problem. Computers & Operations Research, 37(11), 1877–1885. https://doi.org/10.1016/j.cor.2009.06.014