A Matheuristic Algorithm for the Multi-Depot Inventory Routing Problem

Abstract

In this paper we solve a practical Multi-Depot Inventory Routing Problem (*MDIRP*). One minimizes routing costs by determining how to serve the customers from different depots, managing their inventory levels to avoid stock-outs. The *MDIRP* optimizes the trade-off between inventory and routing decisions in an integrated way. We formulate this problem as a mixed-integer linear programming model and design a three-phase matheuristic to solve the problem. The solutions of our matheuristic are compared with those from a branch-and-cut algorithm on classical IRP instances, new instances and a real case study, showing to be very effective.

Keywords: Multi-Depot Inventory Routing Problem, Mixed-Integer Linear Programming, Matheuristic, Clustering.

1. Introduction

In the last decades companies increased their interest in optimizing supply chains. The spread of globalization and the development of information and communication technologies stimulated research towards the development of integrated logistics models, with the aim of reducing the total cost, thanks to a better coordination of the operations. Static and dynamic optimization models have been formulated and solved. Dynamic models are able to take advantage of the availability of online data from information systems to reduce the total logistic cost, e.g.monitoring the inventory levels of the customers in real time. The main contributions are devoted to develop optimization models focused on two or more sequential logistic activities in the supply chain, such as inventory + routing, production + inventory + routing, location + routing, etc. The resulting optimization models are based on two–echelon networks, in which one or more depots deliver products to many customers. In this setting, the *Vendor-Managed Inventory* (*VMI*) paradigm gained importance in different companies in the last decades, thanks to the availability of information in real time. The development of the *VMI* paradigm derives from successful industrial applications in different fields: Procter & Gamble and Walmart (Barratt and Oliveira [1]), chemical products (Dauzère-Pérès et al. [2]), oil and gas (Bell et al. [3], Grønhaug et al. [4], Shen et al. [5]), fuel (Popović et al. [6]) and maritime cement transportation (Christiansen et al. [7]). For an in depth overview of *VMI* applications, we refer to Andersson et al. [8].

Under a *VMI* strategy, the supplier monitors the inventory levels of each retailer and decides on a replenishment policy. The supplier is the leading actor in the decisional process, in order to establish when and how much to deliver, and the routes of the vehicles. This system applies a win-win strategy, because it guarantees an overall reduction of the logistic cost for the supplier and savings in the ordering cost for the customers.

The optimization problem that integrates *VMI* with routing is the well–known *Inventory Routing Problem* (*IRP*). In comparison to the classical *Vehicle Routing Problem* (*VRP*), the *IRP* shows an added complexity due to the integration of the inventory component into a multi-period decisional process. An *IRP* can deal with minimizing the routing cost only or the sum of inventory and routing costs during the planning horizon, while avoiding stock–outs at the customers (see Bertazzi et al. [9] for an introduction to these two classes of *IRPs*). The inventory cost at the supplier and/or at the customers appears in the objective function if the central decision–maker is responsible for it (see Campbell and Salvesbergh [10]). Even if only the routing cost is minimized, this problem is different than the *VRP*, as the quantity to deliver to each customer at each time period is a decision variable in the *IRP*, while this quantity is known in the *VRP* and it is given for a single time period only. In fact, while in the *VRP* the only decision is the set of routes to travel to deliver a given quantity to each customer in a single time period, three main decisions have to be taken in any *IRP*: when and how much to deliver to each customer and the routes to travel at each time period. Moreover, any feasible solution must guarantee that no stock–out occurs over time.

We study an *IRP* in which the deterministic demand of a set of customers is satisfied from a set of depots over a time horizon. These depots are supplied on each day of the quantity needed to face the deliveries on that day. The inventory levels of the customers are managed by the supplier, who is responsible for avoiding stock–out at the customers. Instead, the corresponding inventory cost is paid by the customers, as the supplier does not own the inventories at the customers, like in Campbell and Salvesbergh [10]. Therefore, the supplier minimizes the routing cost only, but guarantees that no stock–out occurs at the customers. The aim is to determine, for each time period, the vehicles to assign to each depot, the customers served by each depot, the quantity to deliver to each customer, and the delivery routes to travel. This problem is referred to as the *Multi–Depot Inventory Routing Problem* (*MDIRP*). The main difference of our problem with respect to the practice implemented in many companies is that the assignment of customers to depots is not fixed a priori, hence we are more flexible. A typical application of this problem is when customers are located in a large urban area. Indeed, when a customer needs a resupply in a high traffic area, the main goal of the supplier is to fulfill (even partially) the request by using vehicles visiting the customer from different depots. In this context the routing problem embedded into the *MDIRP* allows split deliveries.

Literature review

We now review the literature on *IRPs* related to our problem, with the aim of highlighting our contribution with respect to the existing research.

The research literature contains a wide range of IRP formulations varying by objective function, inventory replenishment strategy, and treatment of demand. We first refer to the tutorials by Bertazzi and Speranza [11] and Bertazzi and Speranza [12] and to the survey by Coelho et al. [13] for an introduction and an overview of these problems.

Deterministic *IRP*s are common both from a practical standpoint and as a research area. The single–product and single–vehicle *IRP* was introduced in Archetti et al. [14]. Exact algorithms were designed by Solyalı and Süral [15], Coelho and Laporte [16], Avella et al. [17], Avella et al. [18], while a very effective matheuristic algorithm for this problem was designed by Archetti et al. [19] (see Bertazzi and Speranza [20] for an introduction to matheuristics in solving *IRP*s). Cyclic *IRPs*, where the final inventory levels are equal to the initial inventory levels, were studied by Aksen et al. [21], Ekici et al. [22], Raa and Aghezzaf [23], Aghezzaf et al. [24], Vansteenwegen and Mateo [25], Raa [26]. Several papers investigate the single–product and multi–vehicle *IRP*, including Coelho et al. [27] and Adulyasak et al. [28]. The multi–product and multi-vehicle *IRP* is studied in Coelho and Laporte [29] and Cordeau et al. [30]. Exact algorithms include branch–and–cut, branch–and–price, and more recently, branch–and–price–and–cut (Desaulniers et al. [31]). A very effective matheuristic is proposed by Archetti et al.

[32]. A decomposition approach to tackle large scale instances of the *IRP* can be found in Campbell and Salvesbergh [10] and in Cordeau et al. [30].

Uncertain *IRPs* have been investigated by Kleywegt et al. [33] and Kleywegt et al. [34], where a Markov Decision Problem is formulated for the case of direct shipping and the general case, respectively; by Solyalı et al. [35], where a robust formulation is provided, by Bertazzi et al. [36], where a stochastic dynamic programming approach is designed, by Coelho et al. [37], where dynamic stochastic rules are provided, and by Bertazzi et al. [38], where a min–max dynamic programming approach is implemented. We refer to Roldan et al. [39] for a survey of *IRP* with stochastic lead times and demands. In most of these problems, the offline exact/heuristic policy obtained by solving the stochastic/min–max dynamic programming model can be used online, where the information on the realization of the demand of the customers is made available by the information system. Even deterministic models can be useful in the stochastic setting. For example, an optimal solution of a deterministic model over a short time horizon can be used in a rolling horizon approach to estimate the cost-to-go in an approximate dynamic programming approach.

Contribution to the literature

In this paper we introduce the *Multi–Depot Inventory Routing Problem* (*MDIRP*), as a new problem in the *IRP* literature. To the best of our knowledge, this problem has never been studied before. We just conducted a preliminary study of this problem in Bertazzi et al. [40], where our aim was to understand the importance of driving the solution with a clustering approach in a branch-and-cut algorithm. We formulated this problem as a mixed-integer linear programming model and used a clustering procedure based on the optimal solution of the classical *Capacitated Concentrator Location Routing Problem* (see Bramel and Simchi-Levi [41]). This preliminary paper allowed us to underline that a clustering approach embedded in a branch-and-cut algorithm is promising for obtaining better solutions than the ones obtained by simply applying the same branch-and-cut algorithm without clustering, in the same time limit. However, this preliminary study clearly showed that a more effective and faster heuristic is necessary.

In this paper, our main contribution is to design a novel methodology to solve the *MDIRP* faster and more effectively. We provide a new cluster–based matheuristic composed of the following three steps. In the first step, a new integer program is used to build clusters of customers on the basis of a new quantitative measure of a customer critical inventory level. In the second step, we introduce the concepts of intra and inter-cluster routes. In the third step, a new route–based mixed-integer linear programming formulation is solved to obtain a feasible solution for the problem. The results are compared with the ones obtained solving the mixed integer linear programming formulation of the problem by applying a branch–and–cut algorithm without clustering in the same time limit, on a large number of benchmark instances. Finally, since a typical application of this problem is city logistics problems where several customers are located in a large urban area, we provide a successful application of our methodology to a real problem in an Asian city, where a very large number of nanostores is served by a set of depots over time.

The remainder of the paper is organized as follows. The problem is formally described in Section 2. In Section 3 a mathematical formulation of the *MDIRP* is presented. The branch–and–cut algorithm is described in Section 4. The matheuristic algorithm is presented in Section 5. In Section 6 extensive computational results are presented and discussed. Finally, conclusions are drawn in Section 8.

2. Problem description

We consider a complete undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. We partition the set V in such a way that $V = P \cup I$, where $P = \{1, 2, \dots, m\}$ is the set of depots that deliver a product to the set of customers $I = \{m + 1, m + 2, \dots, m + n\}$ over a finite and discrete time horizon H. Let $T = \{1, 2, \dots, H\}$ be the corresponding set of time periods. The edge set E is defined as $\{(i, j) : i \in I, j \in I, i < j \wedge i \in P, j \in I\}$. A non–negative cost c_{ij} is associated with each edge $(i, j) \in E$. We assume that G is an Euclidean graph, so the triangular inequality holds. A set $K = \{1, 2, \dots, M\}$ of vehicles with capacity C is available to perform deliveries. Each vehicle $k \in K$ can be assigned to one depot $p \in P$ for each time period $t \in T$. Each customer $i \in I$ has a demand d_{it} in period t and can be served by different vehicles from different depots. Two types of split delivery are permitted in the same period: a customer is visited by two vehicles from the same depot or by two vehicles from different depots. Both situations typically occur only if the capacity of each vehicle is small. Moreover, a maximum inventory level U_i and a given starting inventory level I_{i0} are associated with each customer

$i \in I$. We assume that both U_i and I_{i0} are integer. The inventory level of each customer i cannot be negative in each time period $t \in T \cup \{H + 1\}$, i.e., stock–out is not allowed. The aim is to determine:

1. the set of vehicles to assign to each depot at each time period,
2. the quantity of product to deliver to each customer at each time period by using each vehicle,
3. the set of routes to travel at each time period,

that minimize the total routing cost over the time horizon.

3. Mathematical formulation

The aim of this section is to provide a mathematical formulation that will be solved by branch–and–cut. In this way we are able to perform a comparison between the solutions' values obtained with the matheuristic algorithm and the ones found by the branch–and–cut. For this purpose, an edge–based formulation related to the routing problem, in which binary variables are associated with the edges of the graph for each vehicle and each period, is presented. We now present a mathematical formulation of the problem, which is adapted from Archetti et al. [14], in which the multi–depot multi-vehicle case is tackled by using binary variables. We introduce the following notation. Let $\delta(S)$ be the set of edges (i, i') incident to the vertices $i \in S \subset V$, with $i' \notin S$ (edge cutset); for the sake of notation, if $S = \{i\}$, we denote the corresponding edge cutset as $\delta(i)$. Let $E(U)$ be the set of edges (i, j) such that $i, j \in U$, where $U \subseteq I$ is a given set of customers. Our mathematical formulation is based on the following variables:

- I_{it}: inventory level at customer i at the end of period t;
- y_{iktp}: quantity to deliver to customer i in period t by vehicle k from depot p;
- x_{ijktp}: binary variable equal to 1 if vehicle k starting from depot p travels directly from vertex i to vertex j in period t (these variables are defined for $(i, j) \in E$);

- z_{iktp}: binary variable equal to 1 if vehicle k visits vertex i from depot p in period t (note that the variable is not defined when vertex i is a depot different from p, because in our problem the possibility to visit a depot starting the tour from another one is not considered);

The mathematical formulation is described by (1)–(13).

$$\min \quad \sum_{(i,j)\in E}\sum_{k\in K}\sum_{t\in T}\sum_{p\in P} c_{ij}x_{ijktp} \tag{1}$$

$$\text{s.t.} \quad I_{it} = I_{it-1} + \sum_{k\in K}\sum_{p\in P} y_{ikt-1p} - d_{it-1} \qquad \forall t \in T\cup\{H+1\}, \forall i \in I \tag{2}$$

$$I_{it} + \sum_{p\in P}\sum_{k\in K} y_{iktp} \le U_i \qquad \forall t \in T, \forall i \in I \tag{3}$$

$$\sum_{i\in I} y_{iktp} \le C\, z_{pktp} \qquad \forall t \in T, \forall p \in P, \forall k \in K \tag{4}$$

$$\sum_{i\in I} y_{iktp} \ge z_{pktp} \qquad \forall t \in T, \forall p \in P, \forall k \in K \tag{5}$$

$$y_{iktp} \le C\, z_{iktp} \qquad \forall i \in I, \forall t \in T, \forall p \in P, \forall k \in K \tag{6}$$

$$\sum_{p\in P} z_{pktp} \le 1 \qquad \forall t \in T, \forall k \in K \tag{7}$$

$$\sum_{j\in I:(j,i)\in E} x_{jiktp} + \sum_{j\in I:(i,j)\in E} x_{ijktp} = 2z_{iktp} \qquad \forall i \in I, \forall t \in T, \forall p \in P, \forall k \in K \tag{8}$$

$$\sum_{(i,j)\in E(S)} x_{ijktp} \le \sum_{i\in S} z_{iktp} - z_{uktp} \qquad \forall S \subseteq I, |S| \ge 2, \forall t \in T, \forall k \in K, \forall p \in P, u \in S \tag{9}$$

$$x_{ijktp}, x_{pjktp}, x_{jpktp} \in \{0,1\} \qquad \forall i,j \in I, \forall t \in T, \forall p \in P, \forall k \in K \tag{10}$$

$$I_{it} \ge 0 \qquad \forall i \in I, \forall t \in T\cup\{H+1\} \tag{11}$$

$$y_{iktp} \ge 0 \qquad \forall i \in I, \forall t \in T, \forall p \in P, \forall k \in K \tag{12}$$

$$z_{iktp} \in \{0,1\} \qquad \forall i \in I, \forall t \in T, \forall p \in P, \forall k \in K. \tag{13}$$

The objective function (1) minimizes the total routing cost. Constraints (2) are inventory conservation constraints, linking the inventory of period t with that of period $t-1$, plus any deliveries, minus the demand, where $d_{i0} = 0$. Constraints (3) guarantee that the maximum inventory level of each customer does not exceed U_i. Constraints (4) ensure that the total quantity loaded on vehicle k departing from depot p in period t does not exceed the capacity C, and, together with constraints (5) guarantee that, if the total quantity is greater than 0, then the corresponding vehicle is used. Constraints (6) enforce that, if a positive quantity of product is delivered to customer i with vehicle k starting from depot p in time period t, then customer i is visited. Constraints (7) impose that each vehicle k can be assigned to at most one depot p in each time period t. Constraints (8) and (9) control the degree of the vertices and prohibit subtours, respectively. Finally, constraints (10)–(13) define the integrality and non-negativity conditions for the variables. Note that constraints (11) also guarantee that no stock-out occurs at each customer.

4. A Branch–and–cut algorithm

We design the following branch-and-cut algorithm to solve the problem. A branch-and-cut algorithm is one that solves a (mixed-)integer programming model by branch-and-bound to which some constraints are dynamically added while creating the branching tree. Such an algorithm is appropriate in our case because constraints (9) are in exponential number and cannot be enumerated but for very small instances with only a handful of nodes. A branch-and-bound algorithm works by relaxing integrality constraints, solving the problem with continuous variables, and iteratively creating two branches for a variable with a fractional solution. We rely on the branch-and-bound algorithm of a commercial solver. Since constraints (9) are too numerous to be added to the model a priori, we relax them and solve the model without any subtour elimination constraint. The initial linear programming relaxation (*LP*) is obtained by removing constraints (9) from the problem formulation (1)–(13), adding the following constraints that guarantee that each customer i is visited at most once from vehicle k in each period t:

$$\sum_{p \in P} z_{iktp} \leq 1 \quad \forall t \in T, \forall k \in K, \forall i \in I, \tag{14}$$

and then by adding only violated subtour elimination constraints for each period, for each vehicle and for each depot. Constraints (14) are redundant in model (1)–(13), but they can increase the value of the corresponding *LP*. Subtour elimination cuts are separated heuristically along the lines of the procedure designed by Ahr [42]. More precisely, the heuristic finds all the connected components in the auxiliary graph induced by all the edges such that $\overline{x}_{ijktp} \geq \epsilon + \tau$, where $\overline{x}_{ijktp}$ is the value of variable x_{ijktp} on edge (i, j) in the current *LP* while $\epsilon \in \{0, 0.25, 0.50\}$ and τ is a tolerance. Whenever a subset of customers S_p not connected to depot p is found, the corresponding subtour elimination constraint is added for $u = argmax_{i \in S_p} \left\{\overline{z}_{iktp}\right\}$, where $\overline{z}_{iktp}$ is the value of variable z_{iktp} in the current *LP*. This heuristic procedure is also applied to any integer solution of the relaxed formulation, so that the best integer solution of the branch–and–cut is connected to the depot. Note that, when the solution of the LP relaxation is integer in variables x and z, then constraints (9) can be violated only if a subset of customers is not connected to a depot. Therefore, all violations of constraints (9) are found by setting $\epsilon = 0$ and computing the connected components on the corresponding auxiliary graph. Therefore, in this case, this separation algorithm is applied with $\epsilon = 0$ only, and it is able to exactly find all the violations. The branching rule to be used is determined by the MIP solver.

In order to improve the quality of the root node lower bound, the following valid inequalities are added to the *LP* described so far:

1. *Priority inequalities:*

$$z_{iktp} \leq z_{pktp}, \qquad \forall i \in I, \forall k \in K, \forall p \in P, \forall t \in T. \tag{15}$$

   These valid inequalities were presented by Archetti et al. [14] to improve the performance of the proposed branch-and-cut for the single–vehicle *IRP*.

2. *Logical inequalities:*

$$x_{ipktp} + x_{piktp} \leq 2\, z_{iktp}, \qquad \forall i \in I, \forall t \in T, \forall p \in P, \forall k \in K \tag{16}$$

$$x_{ijktp} \leq z_{iktp}, \qquad \forall i, j \in I, \forall t \in T, \forall p \in P, \forall k \in K. \tag{17}$$

   These inequalities are inspired by the logical cuts of Fischetti et al. [43] and Gendreau et al. [44]. Inequalities (16) impose that if the depot p is the predecessor or the successor of customer i in the route executed in period t by vehicle k starting from depot p, then customer i has to be visited in period t by vehicle k starting from depot p. Inequalities (17) impose that if customer j is the successor of customer i in the route performed in period t by vehicle k starting from depot p, then customer i has to be visited in period t by vehicle k starting from depot p.

3. *Disaggregate parity inequalities:*

$$\sum_{(i,j)\in\delta(S)\setminus(F)} x_{ijktp} \geq \sum_{(i,j)\in(F)} x_{ijktp} - |F| + 1,$$
$$\forall t \in T, \forall p \in P, \forall k \in K, \forall F \subseteq \delta(S), |F|\; odd \tag{18}$$

Note that, since split delivery is allowed, aggregate parity inequalities over vehicles are not valid. For example, consider the feasible solution in which, given t and p, customers 1, 2 and 3 are visited in the route traveled by vehicle 1. Moreover, suppose that a direct delivery is used to serve customer 1 from the same depot with vehicle 2. This feasible solution does not satisfy the aggregate parity inequality over vehicles 1 and 2 with $S = \{1\}$ and $F = \{(p, 1)\}$.

4. *Depot–Aggregate parity inequalities:*

$$\sum_{p\in P}\sum_{(i,j)\in\delta(S)\setminus(F)} x_{ijktp} \geq \sum_{p\in P}\sum_{(i,j)\in(F)} x_{ijktp} - |F| + 1,$$
$$\forall t \in T, \forall k \in K, \forall F \subseteq \delta(S), |F|\ odd \tag{19}$$

These inequalities are added dynamically to the *LP* in each node of the branch–and–cut algorithm. Parity inequalities were introduced by Barahona and Grötschel [45] as co–circuit inequalities. They are effective for problems with binary variables, in case that the parity of vertices is required. An example of application of these inequalities is presented by Padberg and Rinaldi [46] for the Symmetric TSP polytope. Inequalities (18) and (19) are separated heuristically according to the procedure described by Aráoz et al. [47].

An initial solution of the branch-and-cut is computed as follows. We solve a relaxed formulation of (1)–(13), named *RMDIRP* in the sequel, in which the routing cost in the objective function is replaced by $\sum_{t\in T}\sum_{k\in K}\sum_{p\in P}\sum_{i\in I} c_{pi}z_{iktp}$, where c_{pi} is the cost of the edge $(p, i) \in E$ connecting the depot p to the customer i. Moreover, all the routing constraints (8) and (9) and the corresponding variables x_{ijktp} are temporarily removed from the mathematical formulation. A set of feasible clusters (one for each vehicle) is defined on the basis of the values of variables z_{iktp} in any optimal *RMDIRP* solution. For each cluster, the optimal TSP tour is computed and the corresponding value of the objective function (1) is then determined. The relaxed formulation is solved repeatedly by adding at every iteration the following diversification constraints, with the aim of obtaining different solutions:

1. Let $\overline{Z}_{ktp} = \left\{i \in I : \bar{z}_{iktp} = 1\right\}$ be the set of the customers served by vehicle k in period t from depot p in the current feasible *RMDIRP* solution. A set of feasible routes is built by solving a TSP on the sub–graph induced by $\overline{Z}_{ktp} \cup \{p\}$, for each k, t and p.

2. The average routing cost is computed over the set of routes returned in step 1. All the routes with a cost greater than the average routing cost are considered as candidates for diversification.

3. For each candidate route, a set of moving nodes, B_{ktp}, is built as follows. The vertices served in the route are ordered according to their non–decreasing insertion cost. The insertion cost of the vertex i is defined as the difference between the optimal TSP tour cost to serve all the customers in the route and the optimal TSP tour cost to serve all the customers in the route except i. The first $\left\lfloor \frac{|\overline{Z}_{ktp}|}{2} \right\rfloor$ vertices are inserted in the set B_{ktp}.

4. The diversification constraint is formulated as follows:

$$\sum_{i \in B_{ktp}} (1 - z_{iktp}) + \sum_{i \in I \setminus B_{ktp}} z_{iktp} \geq \left\lceil \frac{|B_{ktp}|}{|P|} \right\rceil. \tag{20}$$

   Observe that the number of customers that can be moved among the routes decreases with the number of depots. Inequalities (20) are added to the $RMDIRP$ and the new problem is re-optimized. For each problem with diversification constraints, a time limit of 20 minutes is imposed.

5. The procedure ends when a maximum number ω of iterations is reached or $GAP = \frac{(c_W - c_B)}{c_B} 100 \geq \vartheta$, where c_W and c_B are the costs of the worst and the best $RMDIRP$ solutions respectively, while ϑ is a gap limit.

At the end of the procedure the best solution found is used as starting solution for the branch–and–cut algorithm. The pseudocode of this procedure is provided in Algorithm 1.

The proposed branch-and-cut algorithm shares the following features with the existing branch-and-cut algorithms designed for IRPs (see Archetti et al. [14], Adulyasak et al. [28], Archetti et al. [48]). Subtours elimination constraints are removed from the formulation and dynamically added, as are priority and logical valid inequalities. The initial upper bound is obtained by applying a heuristic algorithm. The main differences are that different separation algorithms are used, specific valid inequalities (disaggregate parity and depot-aggregate parity) are used, while symmetry breaking constraints and fractional capacity cuts are not added. Finally, the branching rule is left to the MIP solver, instead of giving priority to the z variables.

Algorithm 1 Initial solution for the branch-and-cut algorithm

```
Set $\kappa = 0$;
Let $\bar{s}_{\mathcal{RMDIRP}}$ be a solution for the $RMDIRP$
Set $Ws_{\mathcal{RMDIRP}} = Bs_{\mathcal{RMDIRP}} = \bar{s}_{\mathcal{RMDIRP}}$; where $Ws_{\mathcal{RMDIRP}} - Bs_{\mathcal{RMDIRP}}$ is the worst/best $RMDIRP$ solution, whose cost is $c_W - c_B$;
Set $GAP = 0$, $k = 0$
while Problem is feasible and $GAP \leq \vartheta$ and $\kappa \leq \omega$ do
    determine the route set $\mathcal{R}^\kappa$;
    determine the average routing cost $c^\kappa_{\mathcal{R}}$ of routes in $\mathcal{R}^\kappa$
    for each route $r \in \mathcal{R}^\kappa$ do
        if $c_r \geq c^\kappa_{\mathcal{R}}$ then
            build set $B_{ktp}$
            add diversification constraint (20)
        end if
    end for
    Solve the $RMDIRP$ with diversification constraints, let $\bar{s}_{\mathcal{RMDIRP}}$ be its solution.
    if $c_{s_{\mathcal{RMDIRP}}} < c_{Bs_{\mathcal{RMDIRP}}}$ then
        Set $Bs_{\mathcal{RMDIRP}} = \bar{s}_{\mathcal{RMDIRP}}$
    end if
    if $c_{s_{\mathcal{RMDIRP}}} > c_{Ws_{\mathcal{RMDIRP}}}$ then
        $Ws_{\mathcal{RMDIRP}} = \bar{s}_{\mathcal{RMDIRP}}$
    end if
    $GAP = \frac{(c_B - c_W)}{c_B} 100$
    $\kappa = \kappa + 1$
end while
```

5. A Matheuristic for the *MDIRP*

We propose a matheuristic algorithm for the solution of the *MDIRP* to solve realistic–size instances. It is based on the following three–phase decomposition approach:

- *Clustering phase*: the idea of grouping customers in a set of clusters is not new. A similar idea is presented by Salhi et al. [49] for a multi–depot multi–vehicle *VRP*. They group customers in two sets: the first one composed of customers served by the depot nearest to them, and the second serving borderline customers. They define borderline customers as those approximately equally closed to two depots. They use this classification as starting point for routes generation. Here, we use the same concept of borderline customers, but we implement a different clustering procedure, strongly focused on the nature of the *MDIRP*. More precisely, an integer linear programming model is solved to generate a partition of the set of customers into a set of clusters, one cluster for each depot $p \in P$.

- *Routing construction phase*: a set $\mathcal{R}$ of routes is built for the clusters generated in the first phase. The routes are generated on the basis of several replenishment policies and assuming different vehicle capacities, yielding very different routes. Two types of routes are generated: *intra-cluster routes* and *inter-cluster routes*. In the first type, each route can only visit customers within the cluster, while in the second type each route can also visit borderline customers. This latter type of routes is introduced to add flexibility in the construction of the routes.

- *Optimization phase*: a binary linear programming model, referred to as *Route-based MDIRP*, is optimally solved to obtain a feasible solution of the *MDIRP*, selecting routes from the set $\mathcal{R}$ and determining delivery quantities to each customer in a given period of the time horizon. This quantity can be different than the one used to generate the routes in the previous phase.

We now describe these three phases in detail.

5.1. Clustering phase

The aim of this phase is to partition the set I of customers into a set of clusters $C = \{C_1, C_2, \ldots, C_{|P|}\}$. We identify *critical customers* and build clusters having two main features: a limited number of customers and a maximum average critical level. The idea to identify critical customers in an *IRP* was proposed by Campbell and Salvesbergh [10], where a qualitative definition of critical customers is proposed. Here, we provide a quantitative measure, computed as the minimum number of deliveries needed to serve the customer over the time horizon, and of the average distance of the customer from the depots. The corresponding critical level CL_i is computed as:

$$CL_i = \alpha R_i + (1 - \alpha) M_i, \tag{21}$$

where α takes values between 0 and 1, while R_i and M_i are respectively the minimum number of deliveries to customer i over the time horizon and the average delivery cost for each customer i with respect to all the depots. These parameters are obtained by computing:

$$\hat{R}_i = \frac{\max\left\{0, \sum_{t \in T} d_{it} - I_{i0}\right\}}{U_i}$$

$$\hat{M}_i = \frac{\sum_{p \in P} DC_i^p}{|P|},$$

where $DC_i^p = c_{TST}^p - c_{TST\backslash\{i\}}^p$ is an estimated transportation cost to serve customer i from depot p, computed as the difference between the optimal TSP tour cost to serve all customers in I from depot $p \in P$ (c_{TST}^p) and the optimal TSP tour cost to serve all customers $I \backslash \{i\}$ from depot $p \in P$ ($c_{TST\backslash\{i\}}^p$). The values of R_i and M_i in (21) are obtained by normalizing $\hat{R}_i$ and $\hat{M}_i$ in the interval $[0, 10]$, where 0 is assumed to be the minimum value and 10 to be the maximum value.

The values of CL_i are the input data of the following binary linear programming model aimed at building the clusters. Let CC be the maximum number of customers for each cluster, TC be the maximum value of the average critical level for each cluster computed with respect to CC, and b_{pi} a binary variable equal to 1 if customer i belongs to the cluster C_p built around depot p, and 0 otherwise.The aim is to create as many clusters as depots. This means, for example, that cluster C_p is the one made by the subset of customers that can be served by routes starting from node p. Then, the model is formulated as follows:

$$\min \quad \sum_{(p,i)\in E: p\in P} c_{pi}\, b_{pi} \tag{22}$$

$$\text{s.t.} \quad \sum_{i\in I} b_{pi} \leq CC \qquad \forall\, p \in P \tag{23}$$

$$\sum_{p\in P} b_{pi} = 1 \qquad \forall\, i \in I \tag{24}$$

$$\sum_{i\in I} CL_i b_{pi}/CC \leq TC \qquad \forall\, p \in P \tag{25}$$

$$b_{pi} \in \{0,1\} \qquad \forall\, p \in P,\ \forall i \in I. \tag{26}$$

The objective function (22) minimizes the total distance between customers and depots. Constraints (23) enforce the total number of customers in each cluster to be within the maximum number CC. Constraints (24) impose that each customer is assigned to exactly one cluster. Constraints (25) enforce the average critical level of each cluster to be not greater than the maximum value TC. Constraints (26) define variables b_{pi}.

In the following, we refer this procedure as the one that receives an instance of the *MDIRP* as input, I_{MDIRP}, and returns a set of clusters $C = \{C_1, \ldots, C_{|P|}\}$, that is: $C \leftarrow Clustering\,(I_{MDIRP})$.

5.2. Routing construction phase

The aim of the second phase is to generate a restricted number of routes for the *MDIRP* based on the clusters generated by model (22)–(26), and on some replenishment policies. Two classes of routes are generated: *intra-cluster* and *inter-cluster routes*. The first class is composed of all the routes serving customers that are in the same cluster, while the second class is composed of all the routes that also serve borderline customers.

Let us first describe the intra-cluster routes we generate. For each cluster C_p, we focus on direct delivery routes from the depot to one customer and on routes built by aggregating the customers served in the same time period based on different replenishment policies. Indeed, the way used to serve a set of customers in the same route is even affected by the replenishment policies of the customers. Therefore, we adopt several resupply policies allowing us to precompute delivery quantities according to different intershipment times. These policies are as follows:

- *Order–up–to level policy.* This policy, inspired by the corresponding policy in Archetti et al. [14], aims at restoring the maximum inventory level at customer i whenever the demand is greater or equal to the current inventory level. The rationale of this policy is to have an inventory level at the customer enough to satisfy the demand of the customer on some days. This policy is referred to as g_1. For each customer i in cluster C_p and for each time period $t \in T$, the replenishment quantity $\hat{y}_{it}(g_1)$ is computed as follows:

$$\hat{y}_{it}(g_1) = \begin{cases} U_i - \hat{I}_{it-1}(g_1), & \text{if } d_{it} \geq \hat{I}_{it-1}(g_1) \\ 0, & \text{otherwise} \end{cases}$$

where $\hat{I}_{it-1}(g_1) = I_{i0} + \sum_{h=1}^{t-2} \hat{y}_{ih}(g_1) - \sum_{h=1}^{t-2} d_{ih}$.

- *Demand level policy.* This policy aims to have an inventory level equal to the demand, whenever the demand is greater or equal to the current inventory level. The rationale of this policy is to have an inventory level at the customer equal to 0 after serving the demand of the customer. This policy is referred to as g_2. For each customer i in cluster C_p and for each time period $t \in T$, the replenishment quantity $\hat{y}_{it}(g_2)$ is computed as follows:

$$\hat{y}_{it}(g_2) = \begin{cases} d_{it} - \hat{I}_{it-1}(g_2), & \text{if } d_{it} \geq \hat{I}_{it-1}(g_2) \\ 0, & \text{otherwise} \end{cases}$$

where $\hat{I}_{it-1}(g_2) = I_{i0} + \sum_{h=1}^{t-2} \hat{y}_{ih}(g_2) - \sum_{h=1}^{t-2} d_{ih}$.

- *Initial–inventory–level policy.* This replenishment policy aims at restoring the maximum between the initial inventory level I_{i0} and the demand d_{it} in each time period, whenever the current inventory level is lower than the demand. The rationale of this policy is to have an inventory level in the interval between the level obtained by applying the policy g_1 and the one obtained by applying the policy g_2, based on the initial inventory level. We refer to this policy as g_3. For each customer i in the cluster C_p and for each time period $t \in T$, the replenishment quantity $\hat{y}_{it}(g_3)$ is computed as follows:

$$\hat{y}_{it}(g_3) = \begin{cases} \max\left\{I_{i0} - \hat{I}_{it-1}(g_3), d_{it} - \hat{I}_{it-1}(g_3)\right\}, & \text{if } d_{it} \geq \hat{I}_{it-1}(g_3) \\ 0, & \text{otherwise} \end{cases}$$

where $\hat{I}_{it-1}(g_3) = I_{i0} + \sum_{h=1}^{t-2} \hat{y}_{ih}(g_3) - \sum_{h=1}^{t-2} d_{ih}$.

- *Critical–customers–level policy.* This replenishment policy tries to match the specific features of the $MDIRP$, as it aims at delivering the demand or twice its value, depending on the value of the critical level CL_i, to the customers not having enough inventory level to satisfy the demand. We refer to this policy as g_4. For each customer i in the cluster C_p and for each time period $t \in T$, the replenishment quantity $\hat{y}_{it}(g_4)$ is computed as follows:

$$\hat{y}_{it}(g_4) = \begin{cases} \min\{2d_{it}, U_i\}, & \text{if } d_{it} \geq \hat{I}_{it-1}(g_4), \text{ and } CL_i \geq 6 \\ d_{it}, & \text{if } d_{it} \geq \hat{I}_{it-1}(g_4), \text{and } CL_i < 6 \\ 0, & \text{otherwise} \end{cases}$$

where $\hat{I}_{it-1}(g_4) = I_{i0} + \sum_{h=1}^{t-2} \hat{y}_{ih}(g_4) - \sum_{h=1}^{t-2} d_{ih}$.

Note that the threshold on the critical level CL_i has been set equal to 6, that is an average value in the scale from 1 to 10.

Each replenishment policy allows us to define a set of customers served in each time period. If the vehicle capacity is respected, then a TSP route is generated starting from the depot and visiting all the served customers. If the capacity constraint is exceeded, some deliveries are moved to the previous or to the next day until feasibility is recovered. More precisely, the following steps are executed to construct a feasible delivery schedule:

- For each period $t = 1, \ldots, H - 1$, if the vehicle capacity is exceeded, the customers to be served in t are sorted in non–decreasing value of the critical level CL_i and, following this ordering, they are moved to time period $t + 1$ until the capacity constraint is satisfied. In $t + 1$ all the replenishment quantities of the postponed customers are re-computed according to the replenishment policy.

- If the capacity constraint is violated in period $t = H$, customers are sorted in non–increasing value of the critical level CL_i and, following this ordering, they are moved to period $t - 1$ until the capacity constraint is satisfied. In $t - 1$ all the delivery quantities of the anticipated customers are re-computed according to the replenishment policy. This procedure is repeated until $t = 2$.

- For each time period, a delivery route is built by solving a TSP on the sub–graph induced by all the customers to be served in that time period and the corresponding depot.

This generation of intra-cluster routes for cluster C_p is executed with three different values of the capacity of the vehicle: C, $C/2$ and $C/3$. This meets the aim of generating different routes. In the following, we refer to this procedure as Intra–cluster Route Generation ($INTRARG$), which operates on the basis of a given time horizon H, replenishment policy g, cluster C_p and capacity Cap, and returns a set of TSP routes over H supplying customers of C_p with policy g, $TSPR^g_{C_p}$, that is: $TSPR^g_{C_p} \leftarrow INTRARG\left(H, g, C_p, Cap\right)$.

Let us now describe how inter-cluster routes are generated. For each cluster C_p:

- Build the corresponding rectangular convex hull, i.e., the smallest rectangular area including all the points associated with the geographical coordinates of the customers in the cluster. Let X^p_{min}, X^p_{max}, Y^p_{min} and Y^p_{max} be the minimum and the maximum values of customers' coordinates, respectively, of the cluster C_p. The corresponding rectangular convex hull is built over the following vertices: (X^p_{min}, Y^p_{min}), (X^p_{min}, Y^p_{max}), (X^p_{max}, Y^p_{min}) and (X^p_{max}, Y^p_{max}). Let DG_p be the diagonal of this rectangular convex hull (see Figure 1).

- Build the set $Bord_p$ of the borderline customers: this set is composed of all the customers not included in C_p whose distance from the nearest extreme vertex of the rectangular convex hull is less than λDG_p, where $0 < \lambda < 1$ is a parameter based on the number of clusters.

- Build the inter-cluster routes serving the customers in the set $Bord_p \cup C_p$: they are built using the same procedure adopted to generate intra–cluster routes, in which only g_1, g_2 and g_3 and capacity Cap equal to C, $C/2$ and $C/3$ are considered (see Figure 1).

In the following, we refer to this procedure as Inter–cluster Route Generation ($INTERRG$), which operates on the basis of a given time horizon H, replenishment policy g, cluster $Bord_p \cup C_p$ and of capacity Cap, and returns a set of TSP routes over H supplying customers of $Bord_p \cup C_p$ with policy g, $TSPR^g_{Bord_p \cup C_p}$, that is: $TSPR^g_{Bord_p \cup C_p} \leftarrow INTERRG\left(H, g, Bord_p \cup C_p, Cap\right)$.

A set $\mathcal{R}$ of delivery routes is built from the union of the sets of intra–cluster routes and inter–cluster routes, excluding all duplicated routes.

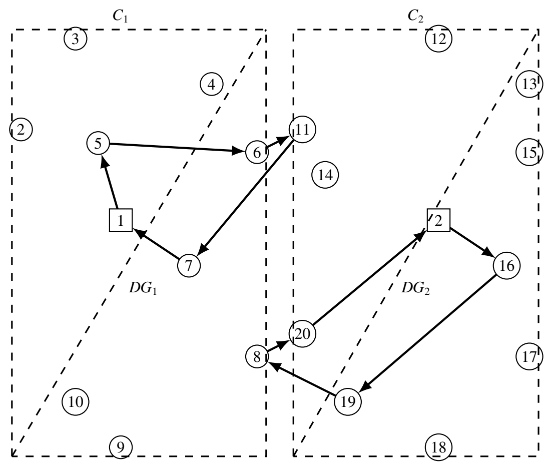

Figure 1: Examples of rectangular convex hull and inter-cluster routes.

5.3. *Optimization phase*

In the third phase, the following route-based *MDIRP*, based on the set of routes $\mathcal{R}$, is solved to determine a feasible solution of the *MDIRP*. We introduce the following parameters that use an explicit index of the routes $r \in \mathcal{R}$:

- c_r: cost of route $r \in \mathcal{R}$
- a_{ir}: binary parameter equal to 1 if customer i is served in route r, 0 otherwise

and the following decision variables:

- m_{irt}: quantity shipped to customer i in route r in time period t
- Inv_{it}: inventory level of customer i at the end of time period $t \in T \cup \{H + 1\}$
- n_{rt}: binary variable equal to 1 if route r is used in time period t and 0 otherwise.

The problem is then formulated as follows:

$$\min \quad \sum_{t\in T}\sum_{r\in\mathcal{R}} c_r\, n_{rt} \tag{27}$$

$$\text{s.t.} \quad Inv_{i,t} = Inv_{it-1} + \sum_{r\in R} m_{irt-1} - d_{it-1}, \qquad \forall i \in I, \forall t \in T \cup \{H+1\} \tag{28}$$

$$\sum_{r\in\mathcal{R}} m_{irt} + I_{it} \leq U_i, \qquad \forall t \in T, \forall i \in I \tag{29}$$

$$\sum_{i\in I} m_{irt} \leq C\, n_{rt}, \qquad \forall r \in \mathcal{R}, \forall t \in T \tag{30}$$

$$m_{irt} \leq C\, a_{ir}, \qquad \forall i \in I, \forall r \in \mathcal{R}, \forall t \in T \tag{31}$$

$$\sum_{r\in\mathcal{R}} n_{rt} \leq M, \qquad \forall t \in T \tag{32}$$

$$Inv_{it} \geq 0, \qquad \forall i \in I, \forall t \in T \cup \{H+1\} \tag{33}$$

$$m_{irt} \geq 0, \qquad \forall i \in I, \forall r \in \mathcal{R}, \forall t \in T \tag{34}$$

$$n_{rt} \in \{0,1\}, \qquad \forall r \in \mathcal{R}, \forall t \in T. \tag{35}$$

The objective function (27) minimizes the total routing cost. Constraints (28) define the inventory level at each customer i at each time period t. Constraints (29) enforce the maximum inventory level of each customer i at each time period t to be not greater than U_i. Constraints (30) enforce the total amount delivered with each route r in time period t to be within the vehicle capacity, while constraints (31) guarantee that a quantity can be delivered to customer i by vehicle r in period t only if customer i is served by route r. Constraints (32) enforce the number of routes used in each time period t to be within the fleet size M. Finally, constraints (33)–(35) define the decision variables.

The overall scheme of the matheuristic algorithm designed for the *MDIRP* is described in Algorithm 2.

In order to better explain the situation described in our problem, an example is introduced in the Figure 2, while Table 1 shows the solution in terms of inventory levels computed at the beginning of each period. From the left to the right we consider the system evolution in terms of inventory levels and delivery quantities during the time horizon. Initially, the customers are assigned to the depots during the clustering phase (they are represented with different colors), then the routing phase is solved for obtaining the final solution. In the example, we consider two depots that have to serve five customers over three periods. Each customer is featured with

Algorithm 2 Matheuristic for the *MDIRP*

```
$\mathcal{R} := \emptyset$
**Phase 1: clustering**
**for** $i \in I$ **do**
    compute parameters $R_i$, $M_i$, $CL_i = \alpha R_i + (1 - \alpha) M_i$
**end for**
solve the **clustering** problem (22)–(26), $C \leftarrow Clustering\,(I_{MDIRP})$
**Phase 2: Routing construction**
**for** each cluster $C_p \in C$ **do**
    **for** each customer $i \in C_p$ **do**
        determine the direct delivery routes $r_i^p$ from depot $p \in C_p$ to $i$
        add this route to $\mathcal{R}$: $\mathcal{R} := \mathcal{R} \cup \left\{r_i^p\right\}$
    **end for**
    **for** each replenishment policy $g_1, g_2, g_3, g_4$ **do**
        **for** each capacity $Cap \in \left\{C, \frac{C}{2}, \frac{C}{3}\right\}$ **do**
            **for** each customer $i \in C_p$ **do**
                **for** each $t \in T$ **do**
                    compute the replenishment quantity $\hat{y}_{it}(g)$;
                **end for**
            **end for**
            determine $TSPR^g_{C_p} \leftarrow INTRARG\left(H, g, C_p, Cap\right)$
            update $\mathcal{R}$: $\mathcal{R} = \mathcal{R} \cup TSPR^g_{C_p}$
            determine $TSPR^g_{Bord_p \cup C_p} \leftarrow INTERRG\left(H, g, Bord_p \cup C_p, Cap\right)$
            update $\mathcal{R}$: $\mathcal{R} = \mathcal{R} \cup TSPR^g_{Bord_p \cup C_p}$
        **end for**
    **end for**
**end for**
**Phase 3: Optimization**
eliminate duplicated routes from $\mathcal{R}$
solve the **route-based** *MDIRP* formulation (27)–(35)
```

an initial inventory level and a demand. The model satisfies the demand and avoids stock-out.

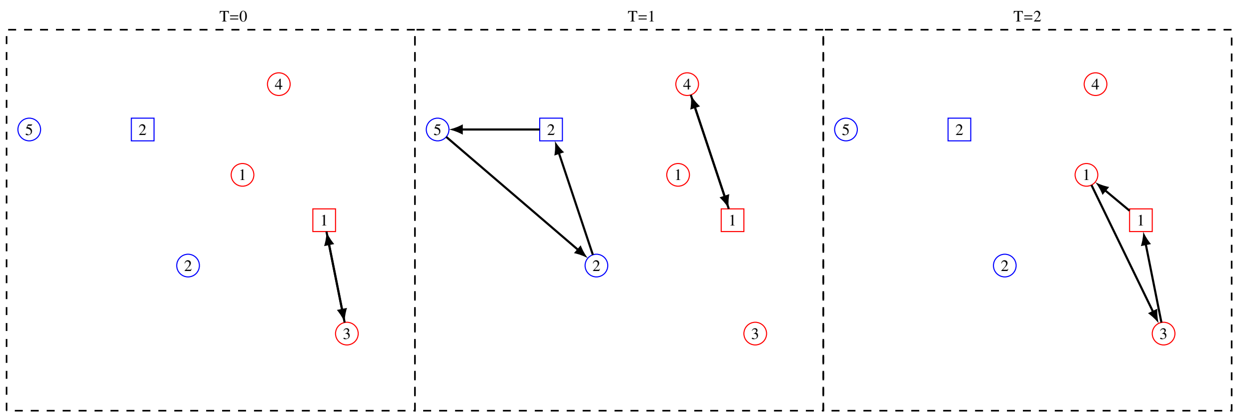

Figure 2: MDIRP system example.

	$T = 0$					$T = 1$					$T = 2$				
Inventory	$I_{10} = 130$	$I_{20} = 70$	$I_{30} = 58$	$I_{40} = 48$	$I_{50} = 11$	$I_{11} = 65$	$I_{21} = 35$	$I_{31} = 58$	$I_{41} = 24$	$I_{51} = 0$	$I_{12} = 0$	$I_{22} = 35$	$I_{32} = 0$	$I_{42} = 24$	$I_{52} = 11$
Demand	$D_{10} = 65$	$D_{20} = 35$	$D_{30} = 58$	$D_{40} = 24$	$D_{50} = 11$	$D_{11} = 65$	$D_{21} = 35$	$D_{31} = 58$	$D_{41} = 24$	$D_{51} = 11$	$D_{12} = 65$	$D_{22} = 35$	$D_{32} = 58$	$D_{42} = 24$	$D_{52} = 11$
Delivery	$Q_{10} = 0$	$Q_{20} = 0$	$Q_{30} = 58$	$Q_{40} = 0$	$Q_{50} = 0$	$Q_{11} = 0$	$Q_{21} = 35$	$Q_{31} = 0$	$Q_{41} = 44$	$Q_{51} = 22$	$Q_{12} = 65$	$Q_{22} = 0$	$Q_{32} = 58$	$Q_{42} = 0$	$Q_{52} = 0$

Table 1: MDIRP solution example.

6. Computational results

The branch-and-cut algorithm described in Section 4 and the matheuristic described in Section 5 were coded in C++ and compiled with g++ -O3. Computational experiments were carried out on a PC equipped with an Intel Core i7-6500U CPU running at 2.50 GHz, with 8 GB of RAM with the Scientific Linux 6.6 operating system. We use the MIP solver IBM CPLEX 12.6.1 using its default settings. To solve the TSP we use the *Concorde TSP Solver* (`http://www.math.uwaterloo.ca/tsp/concorde.html`) that provides optimal solutions for TSP instances with thousands of nodes, and only if Concorde fails to provide an optimal solution within 5 seconds, we call the LKH v3 library (`http://akira.ruc.dk///~keld/research/LKH/`) which solves the TSP heuristically to near optimality.

The performances of the algorithms are evaluated on two different sets of instances. The first one is the benchmark set of the Single-Depot Single-Vehicle *IRP* instances from Archetti et al. [14]; in this case the results obtained applying the three–phase matheuristic algorithm were compared with the optimal results available in literature. The second set of instances is specific for the multi–depot and multi–vehicle case, and it is derived from the same instances. Specifically, we maintain the original depot as the first depot and we generate the remaining $p-1$ depots randomly. Our data set is composed of 100 instances with 5 to 50 customers. For each number of customers, five instances are generated with the number of depots from 2 to 6 according to the size of the instance. The time horizon H is 3 and 6. In order to generate multi–vehicle instances, we consider a fleet of 3 vehicles with a capacity that is reduced by $\frac{1}{3}$ up to 1, with respect to the capacity in the original instances. Note that the clustering problem (22)-(26) and the route-based *MDIRP* (27)-(35) are defined through a polynomial number of variables and constrains, so they are solved using the commercial MIP solver IBM CPLEX 12.6.1. Because there are no optimal solutions for the new data set of instances designed for the *MDIRP*, the solution values provided by the matheuristic are compared with the best upper bounds obtained with the branch–and–cut described in Section 4.

6.1. Matheuristic performance on the Single-Depot, Single-Vehicle IRP

The matheuristic algorithm is executed on the classical *IRP* instances for the single-depot and single-vehicle problem. This data set is composed of 100 instances with up to 50 customers, a time horizon $H = 3, 6$ and inventory cost h_i in $[0.1, 0.5]$. Instances are labelled as $absXnY$, where X is the instance number and Y is the customer number. The data related to the supplier and its inventory costs are not considered. The goal is to demonstrate that the matheuristic, designed for a the multi-depot and multi-vehicle case, also performs well on Single-Depot Single-Vehicle *IRP*.

Numerical results are shown in Table 2. The average results grouped for instance size are described, each group is composed of 5 instances. This table is organized as follows. Column **Instance** provides the instance name. Column **TimeMH (s)** provides the matheuristic computational time, while column **TimeEX (s)** reports the computational time of the exact approach. Column **Gap %** provides the gap between the two approaches, computed as $GAP\ \% = \frac{MH-EX}{EX}100$. These results were provided by Cordeau et al. [30]. Finally columns **Optimal MH** and **Optimal EX** describe the number of instances closed to optimaly into each group for the two approaches. Note that each group is made up of five instances.

The results for the clustering and route generation phases are not reported in the table. Obviously, the first phase is not useful in this configuration, because there is only one cluster related

H=3						H=6					
Instance	**TimeMH (s)**	**TimeEX (s)**	**Gap %**	**Optimal MH**	**Optimale EX**	**Instance**	**TimeMH (s)**	**TimeEX (s)**	**Gap %**	**Optimal MH**	**Optimale EX**
absNn5	0.11	1.09	0.00	5\5	5\5	absNn5	0.63	185.97	0.72	2\5	5\5
absNn10	0.40	2.22	0.00	5\5	5\5	absNn10	1.41	17.07	0.87	1\5	5\5
absNn15	0.75	4.71	1.80	2\5	5\5	absNn15	3.53	7.09	3.31	0\5	5\5
absNn20	1.15	9.22	0.54	4\5	5\5	absNn20	8.81	42.17	2.43	0\5	5\5
absNn25	1.88	14.25	1.43	3\5	5\5	absNn25	8.08	75.31	1.69	0\5	5\5
absNn30	2.85	26.29	2.14	3\5	5\5	absNn30	48.57	458.36	5.56	0\5	5\5
absNn35	3.84	32.82	2.37	4\5	5\5	absNn35	51.36	1402.34	4.05	0\5	5\5
absNn40	6.00	65.28	1.07	3\5	5\5	absNn40	67.74	3994.09	3.17	0\5	5\5
absNn45	6.09	100.33	0.61	4\5	5\5	absNn45	80.65	6471.17	2.39	0\5	4\5
absNn50	11.98	208.81	0.54	1\5	5\5	absNn50	70.35	18709.67	2.76	0\5	3\5
Average	3.50	46.5	1.04			Average	34.11	3131.32	2.69		

Table 2: Performance on the Single-Depot, Single-Vehicle *IRP*

to the single depot. The route generation phase is very fast and effective. On average 42 routes are generated with $H = 3$, and 62 with $H = 6$. In the instances with $H = 3$, the matheuristic provides an optimal solution for a large number of instances with a computational time that is much smaller than the one required by the exact solver. The average duality gap is only 1.04%. In the instances with $H = 6$, the number of optimal solution provided is smaller than the previous case, but the average duality gap is around 2.69%. The savings in computational time are higher in this case. We can conclude the matheuristic is very effective to solve the Single-Depot, Single-Vehicle *IRP*.

6.2. Matheuristic performance on the Multi-Depot, Multi-Vehicle IRP

In this section the results for the *MDIRP* are presented. Firstly, the three phase matheuristic is tested on the modified set of instances derived from Archetti et al. [14], as previously explained. Secondly, the branch–and–cut algorithm is executed on the same data set of instances and the best upper bounds found by the exact method are compared with the values of the solutions provided by the matheuristic. The instances are labelled as *SnNdDhH*, where S indicates the instance number, N is the number of customers in the instance, D is the number of depots, and H is the number of periods. A time limit of 3 hours was imposed to CPLEX for solving the mathematical model given by (27)–(35), while the branch-and-cut was run with a time limit of 6 hours. The following parameters are set after preliminary tests on a subset of instances: $\epsilon = 0.2$, $\omega = 10$, $\alpha = 0.2$, $TC = 6$, $CC \in \{3, \ldots, 15\}$ and $\lambda \in \{0.2, \ldots, 0.5\}$.

Tables 3 shows the results provided by the matheuristic algorithm. These tables are organized as follows. Column **Instance** denotes the instance name. Column **N. Clust.** reports the number of generated clusters, while column $|C_i|$ reports the cardinality of each cluster. Column **N. Routes** provides the number of intra–cluster and inter–cluster routes. The computational time of the algorithm is reported in column **Time (s)**, while the cost of the best solution is reported in column **Cost**. Finally, column **MHIRP GAP%** shows the gap provided by CPLEX to solve model (27)–(35). Unlike the exact method, our matheuristic provided solutions within a few minutes of computing time. The results show that clusters are balanced in terms of number of customers and that the number of generated routes is very small with respect to the potential number of routes. The average number of routes that are generated is 462 for the set with $H = 3$, and is 492 for the set with $H = 6$. The number of routes exceeds 1000 in only 6% of the instances, the ones with the largest number of customers. The average computational time is 88 seconds for the data set with $H = 3$, while is around 1965 seconds for the data set with $H = 6$.

6.2.1. Branch-and-cut performance

Table 4 provides average results for the branch–and–cut algorithm for the instances with $H = 3$ and $H = 6$, respectively. The average results are grouped for instance size, each group is composed of 5 instances. The tables are organized as follows. Column **Instance** provides the instance name. Column **Time (s)** shows the computational time, while column **Nodes** provides the number of nodes processed in the branch-and-cut algorithm. Columns **Subtours**, **Dis Parity Ineqs** and **Aggr Parity Ineqs** report the number of added sub–tour elimination, disaggregate parity and aggregate parity inequalities that are added dynamically to the *LP*, respectively. Finally, column **Gap (%)** provides the optimality gap. These results show that, even if an initial solution is provided and several families of cuts are used, the problem remains very challenging to be solved exactly. Even for small instances with 10 customers and 3 periods, optimality is not achieved for all instances. Several instances with more than 35 customers used all the computational time at the root node only, and the average gap was 31.24% for instances with 3 periods and 36.95% for those with 6 periods. The first reason why our branch-and-cut does not perform well in the Multi-Vehicle and Multi-Depot *IRP* can be attributed to the fact that the multi-vehicle case allows symmetry solutions when the fleet of vehicles is homogeneous. This makes the problem much more difficult to be solved, even if symmetry breaking inequalities are added to the initial LP. On the other hand, the quality of the lower bound corresponding to the initial LP is very poor, and the addition of the symmetry breaking inequalities and of some

Instance	N.Clust.	$\lvert C_i\rvert$	N. Routes	Time (s)	Cost	MHIRP GAP %	Instance	N. Clust.	$\lvert C_i\rvert$	N. Routes	Time (s)	Cost	MHIRP GAP%
1n5d2h3	2	4,3	10	0.09	1148.80	0.00	1n5d2h6	2	4,3	19	0.17	2595.14	0.00
2n5d2h3	2	4,3	16	0.1	956.24	0.00	2n5d2h6	2	4,3	15	3.76	3296.32	0.00
3n5d2h3	2	3,4	13	0.13	1801.02	0.00	3n5d2h6	2	4,3	18	0.8	5145.63	0.00
4n5d2h3	2	3,4	12	0.12	1425.39	0.00	4n5d2h6	2	4,3	26	6.59	2963.12	0.00
5n5d2h3	2	3,4	10	0.08	1808.4	0.00	5n5d2h6	2	4,3	26	86.09	3117.05	0.00
1n10d2h3	2	5,7	71	1.76	2112.53	0.00	1n10d2h6	2	5,7	81	11.5	4363.89	0.00
2n10d2h3	2	5,7	75	2.21	2425.97	0.00	2n10d2h6	2	5,7	86	369.07	5271.25	0.00
3n10d2h3	2	8,4	65	0.83	1651.54	0.00	3n10d2h6	2	8,4	82	64.52	5290.70	0.00
4n10d2h3	2	7,5	59	0.81	2449.30	0.00	4n10d2h6	2	4,8	115	665.89	5649.83	0.00
5n10d2h3	2	7,5	62	0.89	1982.90	0.00	5n10d2h6	2	4,8	135	565.2	5052.26	0.00
1n15d2h3	2	7,10	188	3.76	3891.56	0.00	1n15d2h6	2	7,10	239	3466.42	8942.68	0.00
2n15d2h3	2	7,10	169	4.3	2436.43	0.00	2n15d2h6	2	10,7	233	1995.81	9072.41	0.00
3n15d2h3	2	9,8	176	3.84	3189.5	0.00	3n15d2h3	2	8,9	260	1693.07	8775.99	0.00
4n15d2h3	2	8,9	180	3.88	2298.16	0.00	4n15d2h6	2	12,5	272	5516.63	8690.27	0.00
5n15d2h3	2	6,11	222	1.93	2329.42	0.00	5n15d2h6	2	9,8	236	473.50	8905.06	0.00
1n20d3h3	3	7,11,5	255	11.66	3095.91	0.00	1n20d3h6	3	5,10,8	297	6614.13	9616.51	0.00
2n20d3h3	3	7,11,5	255	6.44	4074.56	0.00	2n20d3h6	3	8,8,7	321	1596.92	8386.92	0.00
3n20d3h3	3	7,11,5	269	5.79	3361.74	0.00	3n20d3h6	3	8,8,7	306	1405.85	10895.99	0.00
4n20d3h3	3	8,10,5	252	7.88	4151.87	0.00	4n20d3h6	3	10,8,5	294	89.36	6826.68	0.00
5n20d3h3	3	5,11,7	273	27.81	4235.63	0.00	5n20d3h6	3	13,9,1	495	1577.32	11343.97	0.00
1n25d4h3	4	6,10,5,8	277	13.97	3354.83	0.00	1n25d4h6	4	15,7,2,5	617	9574	11276.2	0.00
2n25d4h3	4	9,7,5,8	263	11.07	3654.25	0.00	2n25d4h6	4	9,7,5,8	397	3728.45	9172.41	0.00
3n25d4h3	4	9,8,6,6	258	13.93	3870.56	0.00	3n25d4h6	4	9,8,6,6	251	92.81	9575.81	0.00
4n25d4h3	4	11,5,8,5	330	43.8	4925.73	0.00	4n25d4h6	4	6,10,7,6	314	37.63	9215.81	0.00
5n25d4h3	4	9,9,9,2	343	9.88	4475.17	0.00	5n25d4h6	4	8,10,6,5	353	538.80	8869.16	0.00
1n30d4h3	4	6,13,5,10	533	76.56	4262.32	0.00	1n30d4h6	4	8,13,7,6	466	10800	12255.30	1.12
2n30d4h3	4	5,14,6,9	562	22.2	3733.09	0.00	2n30d4h6	4	6,11,6,11	545	1575.14	12001.80	0.00
3n30d4h3	4	7,13,4,10	533	18.94	4649.11	0.00	3n30d4h6	4	8,11,4,11	639	2542.20	10675.78	0.00
4n30d4h3	4	11,11,1,11	608	18.26	3580.22	0.00	4n30d4h6	4	11,11,6,6	531	600.42	9875.90	0.00
5n30d4h3	4	11,13,5,5	560	21.9	4199.74	0.00	5n30d4h6	4	11,11,6,6	536	2841.61	9325.37	0.00
1n35d5h3	5	8,9,6,4,13	529	21.98	5351.66	0.00	1n35d5h6	5	9,10,10,4,7	501	61.94	11029.50	0.00
2n35d5h3	5	11,7,7,4,11	531	28.02	5213.10	0.00	2n35d5h6	5	10,9,7,4,10	605	76.15	11195.46	0.00
3n35d5h3	5	7,10,6,6,11	483	27.83	4790.49	0.00	3n35d5h6	5	8,10,6,6,10	651	1877.39	10747.05	0.00
4n35d5h3	5	11,10,5,3,11	581	31.33	5430.31	0.00	4n35d5h6	5	10,8,10,6,6	435	71.3	10389.20	0.00
5n35d4h3	5	11,11,4,1,13	734	32.59	4545.17	0.00	5n35d5h6	5	10,9,11,5,5	607	714.18	14750.82	0.00
1n40d5h3	5	11,4,3,17,10	1085	58.14	5810.59	0.00	1n40d5h6	5	11,8,11,10,5	763	1185.19	12152.40	0.00
2n40d5h3	5	10,7,8,10,10	598	57.39	5543.21	0.00	2n40d5h6	5	10,7,8,10,10	812	4330.62	13797.56	0.00
3n40d5h3	5	13,4,5,13,10	836	70.95	5568.42	0.00	3n40d5h6	5	10,5,10,10,10	865	4939.05	10840.55	0.00
4n40d5h3	5	13,4,7,13,8	798	49.24	5112.32	0.00	4n40d5h6	5	14,8,10,6,7	790	124.50	12337.50	0.00
5n40d5h3	5	10,5,10,10,10	637	98.69	4695.32	0.00	5n40d5h6	5	14,5,16,5,5	1149	2672.15	12673.98	0.00
1n45d6h3	6	15,4,3,15,2,12	654	136.24	6547.97	0.00	1n45d6h6	6	7,13,8,11,6,6	761	576.78	10940.30	0.00
2n45d6h3	6	10,7,4,14,6,10	816	149.27	5896.87	0.00	2n45d6h6	6	10,7,4,10,10,10	713	323.05	11323.10	0.00
3n45d6h3	6	10,4,7,10,10,10	683	69.24	6084.01	0.00	3n45d6h6	6	7,8,8,13,7,8	680	640.15	11606	0.00
4n45d6h3	5	10,3,9,13,6,7	1059	47.41	6420.13	0.00	4n45d6h6	6	8,11,11,10,5,6	701	2236.47	11838.5	0.00
5n45d6h3	6	13,5,7,13,7,6	821	76.56	5002	0.00	5n45d6h6	6	9,13,2,13,6,8	964	2836.15	12392.35	0.00
1n50d6h3	6	7,16,16,5,2,10	1432	2047.59	7238.44	0.00	1n50d6h6	6	13,6,13,11,6,7	1049	5049.94	12892.30	0.00
2n50d6h3	6	7,7,16,8,3,15	1302	197.92	6131.18	0.00	2n50d6h6	6	10,10,10,8,8,10	954	1013.1	14529.04	0.00
3n50d6h3	6	10,10,10,8,8,10	720	61.17	6788	0.00	3n50d6h6	6	10,10,10,8,8,10	918	2778.32	13245.80	0.00
4n50d6h3	6	8,7,13,4,11,13	999	70.88	6283.02	0.00	4n50d6h6	6	16,11,16,2,4,7	1511	13186.52	13186.52	0.00
5n50d6h3	6	9,7,13,8,7,12	893	362.70	5914.12	0.00	5n50d6h6	6	11,8,13,10,6,8	939	4175.72	12588.10	0.00
Average			461.82	87.78		0.00	Average			492	1965.4		0.00

Table 3: Matheuristic performance for instances with $H = 3, 6$

classes of already known valid inequalities does not improve so much this bound. Therefore, closing the gap remains a very difficult issue for the branch–and–cut.

H=3							H=6						
Instance	**Time (s)**	**Nodes**	**Subtours**	**Dis Par. Ineqs**	**Aggr Par. Ineqs**	**GAP %**	**Instance**	**Time (s)**	**Nodes**	**Subtours**	**Dis Par. Ineqs**	**Aggr Par. Ineqs**	**GAP %**
Nn5d2h3	496.50	2837	107	221	158	0.00	Nn5d2h6	19097	15378	400	716	486	10.32
Nn10d2h3	9596.61	3376	566	482	323	2.60	Nn10d2h6	21700	2100	1429	1477	743	30.20
Nn15d2h3	18531.5	2689	1309	951	628	16.99	Nn15d2h6	21700	724	1727	1586	718	31.46
Nn20d3h3	21700	470	1501	1497	720	19.77	Nn20d3h6	21700	40	1496	2294	1077	41.55
Nn25d4h3	21700	86	1431	1667	710	39.99	Nn25d4h6	21700	4	1348	1462	692	43.19
Nn30d4h3	21700	29	1194	1736	747	36.29	Nn30d4h6	21700	1	998	1922	911	43.40
Nn35d5h3	21700	4	919	1281	442	46.15	Nn35d5h6	21700	1	735	936	612	40.33
Nn40d5h3	21700	2	1185	1719	706	51.99	Nn40d5h3	21700	1	784	956	596	40.49
Nn45d6h3	21700	1	871	1476	727	44.88	Nn45d6h6	21700	0	455	572	469	41.76
Nn50d6h3	21700	1	553	1384	676	53.76	Nn50d6h6	21700	1	233	482	239	46.40
Average	18052.46	949	963	1241	584	31.24	Average	21439	1824	961	1240	654	36.95

Table 4: Branch–and–cut performance for instances with $H = 3, 6$

6.2.2. *Performance comparison*

In this section the comparison between the results of the matheuristic algorithm and the branch–and–cut is presented. Table 5 provides the results for all the instances with time horizon $H = 3$ and $H = 6$. The table is organized as follows. Column **Instance** provides the instance name. Columns **MH** and **TimeMH (s)** provide the cost of the matheuristic solution and the corresponding computational time, respectively. Columns **B&C** and **TimeB&C (s)** report the cost of the best branch–and–cut solution obtained within the time limit allowed and the corresponding computational time, respectively. Finally, column **Gap %** provides the gap between the two approaches, computed as $GAP\ \% = \frac{MH-B\&C}{B\&C}100$.

The results shown in Table 5 demonstrate that the matheuristic algorithm is significantly more effective than the branch–and–cut. It is clear that instances with a longer time horizon are more difficult to solve. Nevertheless, the results provided by the matheuristic are good for both data sets. The result demonstrates that the route generation phase is not affected by the time horizon dimension. For this reason, it is possible to solve the instances with $H = 6$ without increasing the number of route–variables in formulation (27)–(35). Instead, the branch–and–cut algorithm can solve only small and medium size instances, while the possibility to find optimal solutions decreases with the instance size. Futhermore, the branch–and–cut algorithm is able to solve to optimality for only 11 small instances. As a consequence, the time limit of 6 hours is reached in 89 instances.

Instance	MH	TimeMH (s)	B&C	TimeB&C (s)	Gap %	Instance	MH	TimeMH (s)	B&C	TimeB&C (s)	Gap %
1n5d2h3	1148.80	0.09	1148.80	89.71	0.00	1n5d2h6	2595.14	0.17	2595.14	8685.15	0.00
2n5d2h3	956.24	0.10	956.24	434.76	0.00	2n5d2h6	3296.32	3.76	3296.32	21700	0.00
3n5d2h3	1801.02	0.13	1801.02	360.51	0.00	3n5d2h6	5145.63	0.80	5145.63	21700	0.00
4n5d2h3	1425.39	0.12	1425.39	1537.42	0.00	4n5d2h6	2963.12	6.59	2963.12	21700	0.00
5n5d2h3	1808.40	0.08	1808.40	60.09	0.00	5n5d2h6	3117.05	86.09	3120.91	21700	-0.12
1n10d2h3	2112.53	1.76	2177.99	21700	-3.00	1n10d2h6	4363.89	11.5	5102.45	21700	-14,47
2n10d2h3	2425.97	2.21	2427.66	21700	-0.06	2n10d2h6	5271.25	369.07	5501.12	21700	-4.18
3n10d2h3	1651.54	0.83	1651.54	1854.78	0.00	3n10d2h6	5290.71	64.52	5617.17	21700	-5.81
4n10d2h3	2449.30	0.81	2449.30	1994.27	0.00	4n10d2h6	5649.83	665.89	5849.66	21700	-3.42
5n10d2h3	1982.90	0.89	1982.90	734	0.00	5n10d2h6	5052.26	565.20	5077.50	21700	-0.50
1n15d2h3	3891.56	3.76	4588.88	21700	-15.20	1n15d2h6	8942.69	3466.42	9904.03	21700	-9.71
2n15d2h3	2436.43	4.30	2436.43	20955	0.00	2n15d2h6	9072.41	1995.81	10202.70	21700	-11.08
3n15d2h3	3189.50	3.84	3075.69	6602.50	3.70	3n15d2h6	8775.07	1693.07	9276.72	21700	-5.41
4n15d2h3	2298.16	3.88	2481.13	21700	-7.37	4n15d2h6	8690.27	5516.63	8875.49	21700	-2.09
5n15d2h3	2329.42	1.93	2912.07	21700	-20.01	5n15d2h6	8905.06	473.50	8997.27	21700	-1.03
1n20d3h3	3095.91	11.66	3263.25	21700	-5.13	1n20d3h6	9616.51	6614.13	11433.40	21700	-15.89
2n20d3h3	4074.56	6.44	4395.27	21700	-7.29	2n20d3h6	8386.92	1596.92	9094.27	21700	-7.78
3n20d3h3	3361.74	5.79	3399.42	21700	-1.11	3n20d3h6	10895.99	1405.85	10807.75	21700	0.82
4n20d3h3	4151.87	7.88	4501.08	21700	-7.76	4n20d3h6	6826.68	89.36	12965.10	21700	-47.34
5n20d3h3	4235.63	27.81	4338.40	21700	-2.37	5n20d3h6	11343.97	1577.32	13545.80	21700	-16.25
1n25d4h3	3354.83	13.97	4203.45	21700	-20.19	1n25d4h6	11276.20	9574	11949.99	21700	-5.64
2n25d4h3	3654.25	11.07	4570.13	21700	-20.04	2n25d4h6	9172.41	3728.45	10369.70	21700	-11.54
3n25d4h3	3870.56	13.93	4127.82	21700	-6.23	3n25d4h6	9575.81	92.81	10511.3	21700	-8.90
4n25d4h3	4925.73	43.80	5722.99	21700	-13.93	4n25d4h6	9215.81	37.63	10457.10	21700	-11.87
5n25d4h3	4475.17	9.88	6214.81	21700	-27.99	5n25d4h6	8869.16	538.70	9556.95	21700	-7.20
1n30d4h3	4262.32	76.56	5206.49	21700	-18.13	1n30d4h6	12255.30	10800	1275.90	21700	-3.94
2n30d4h3	3733.09	22.20	4805.12	21700	-22.31	2n30d4h6	12001.80	1575.14	12848.94	21700	-6.59
3n30d4h3	4649.11	18.94	5344.53	21700	-13.01	3n30d4h6	10675.78	2542.20	11921.20	21700	-10.45
4n30d4h3	3580.22	18.26	4805.12	21700	-25.49	4n30d4h6	9875.90	600.42	11420.40	21700	-13.52
5n30d4h3	4199.74	21.90	4693.64	21700	-10.52	5n30d4h6	9325.37	2841.61	9960.20	21700	-6.37
1n35d5h3	5351.66	21.98	7013.75	21700	-23.70	1n35d5h6	11029.50	61.94	12924	21700	-14.66
2n35d5h3	5213.10	28.02	6138.55	21700	-15.08	2n35d5h6	11195.46	76.15	11561.10	21700	-3.16
3n35d5h3	4790.49	27.83	6228.25	21700	-23.08	3n35d5h6	10747.05	1877.39	11676.1	21700	-7.96
4n35d5h3	5430.31	31.33	6660.30	21700	-18.47	4n35d5h6	10389.2	71.3	11004.45	21700	-5.59
5n35d5h3	4545.17	32.59	5771.25	21700	-21.24	5n35d5h6	14750.82	714.18	15450.90	21700	-4.53
1n40d5h3	5810.59	58.14	7080.27	21700	-17.93	1n40d5h6	12152.40	1185.19	12208.50	21700	-0.46
2n40d5h3	5543.21	57.39	7128.72	21700	-22.24	2n40d5h6	13797.56	4330.62	1399.70	21700	-1.44
3n40d5h3	5568.42	70.95	6434.14	21700	-13.45	3n40d5h6	10840.55	4939.05	10887.50	21700	-0.43
4n40d5h3	5115.32	49.24	5742.3	21700	-10.92	4n40d5h6	12337.50	124.50	12554.70	21700	-1.73
5n40d5h3	4695.32	98.69	5196.53	21700	-9.65	5n40d5h6	12673.98	2672.15	13283.70	21700	-4.58
1n45d6h3	6547.97	136.24	7272.02	21700	-9.96	1n45d6h6	10940.30	576.78	10884.90	21700	0.51
2n45d6h3	5896.87	149.27	6634.67	21700	-11.12	2n45d6h6	11323.10	323.05	11596.40	21700	-2.53
3n45d6h3	6084.01	69.24	6721.43	21700	-9.48	3n45d6h6	11606	640.15	11639.20	21700	-0.29
4n45d6h3	6420.13	47.41	6292.32	21700	2.03	4n45d6h6	11838.5	2236.47	12166.64	21700	-2.29
5n45d6h3	5002	76.46	5650.86	21700	-11.48	5n45d6h6	12392.35	2836.15	13373.20	21700	-7.33
1n50d6h3	7238.44	2407.59	7881.37	21700	-8.16	1n50d6h6	12892.30	5049.94	14031.90	21700	-8.12
2n50d6h3	6131.18	197.92	7432.50	21700	-17.51	2n50d6h6	14529.04	1013.1	14635.36	21700	-0.73
3n50d6h3	6788	61.17	7611.76	21700	-10.82	3n50d6h6	13245.80	2778.32	13721.20	21700	-3.46
4n50d6h3	6238.02	70.88	7740.36	21700	-18.83	4n50d6h6	13122.22	4054.4	13202.47	21700	-0.60
5n50d6h3	5914.12	362.70	6509.11	21700	-9.14	5n50d6h6	12588.10	4175.72	13676.20	21700	-7.96
Average		87.78		18052.46	-10.47	Average		1965.40		21439.75	-5.98

Table 5: Performance comparison

In the set with $H = 3$, the average duality gap is equal to 31.24%, while in the set with $H = 6$ this gap is equal to 36.22%. There is no sensitive difference between the gaps of the two data sets. This is due to the good quality of the initial solution built with the procedure described in Section 4, and used as starting solution for the branch–and–cut algorithm. The availability of this good initial solution allows to reduce the number of nodes to explore in the branch–and–cut tree, mainly in the data set with $H = 6$. The drawback of the branch–and–cut is that in large instances all the computational time is spent adding cuts to the LP formulation at the root node. In 96% of all the cases, the matheuristic is able to find a solution in a smaller computational time than the one provided by the branch–and–cut algorithm. For the data set with $H = 3$, the matheuristic provides a feasible solution with a total cost lower than that of the solution found by the branch–and–cut by 10.47%, on average. The matheuristic is able to find the best solution within a computational time that is 17964.68 seconds smaller on average than the ones of the branch–and–cut. For the data set with $H = 6$, the matheuristic finds a feasible solution that is better by 5.98% on average than the best solution provided by the branch–and–cut, and in a computational time that is 19474.7 seconds smaller on average than the ones of the branch–and–cut.

6.2.3. Impact of the clustering and route generation phases on the quality of the solutions

We now assess the impact of the clustering and route generation phases on the final solution of our matheuristic. The following results highlight the key role played by the inter–cluster routes that are built around the so called "borderline customers", indeed, a tailored clustering procedure and the corresponding intra–cluster routes are not sufficient to guarantee good quality of the solutions. The next table provide the results of the sensitivity analysis in terms of total number of generated routes, total number of inter-cluster and intra-cluster routes, and number of selected intra-cluster and inter-cluster routes. The average results are grouped for instance size, each group is composed of 5 instances. The tables are organized as follows: column **Instance** gives the name of the instance, column **N. Clust.** shows the number of clusters, column **Tot.Routes** reports the total number of generated routes, columns **Intra-cluster R.** and **Inter-cluster R.** provide the total number of intra–cluster and inter–cluster routes, respectively, column **Solution Routes** describes the number of selected routes in the final solution, columns **Sol.intra-c** and **Sol.inter-c** describe the number of selected intra–cluster and of inter–cluster routes, respectively. Finally, column **Incidence** refers to the ratio $\frac{Sol.inter-c}{Solution\ Routes}$.

H=3								
Instance	**N.Clust.**	**Tot.Routes**	**Intra-cluster R.**	**Inter-cluster R.**	**Solution Routes**	**Sol.Intra-c**	**Sol.Inter-c**	**Incidence**
Nn5d2h3	2	12	7	5	4	4	0	0.00
Nnd210h3	2	67	51	16	4	4	1	0.09
Nn15d2h3	2	187	168	17	5	4	1	0.13
Nn20d3h3	3	261	228	33	5	4	1	0.22
Nn25d4h3	4	294	248	46	6	4	2	0.27
Nn30d4h3	4	559	514	45	6	5	1	0.12
Nn35d5h3	5	572	510	61	6	5	1	0.11
Nn40d5h3	5	791	726	65	6	5	1	0.19
Nn45d6h3	6	807	734	73	6	4	2	0.28
Nn50d6h3	6	1069	995	75	7	5	1	0.21
average	4	462	418	44	5	4	1	0.16
H=6								
Nn5d2h6	2	21	9	11	6	6	0	0.00
Nnd210h6	2	100	64	36	9	7	1	0.16
Nn15d2h6	2	248	204	44	9	7	1	0.16
Nn20d3h6	3	343	279	64	10	9	1	0.13
Nn25d4h6	4	386	308	78	12	10	2	0.14
Nn30d4h6	4	543	447	96	12	10	2	0.16
Nn35d5h6	5	560	463	97	13	8	5	0.36
Nn40d5h6	5	876	745	131	14	11	2	0.17
Nn45d6h6	6	764	639	125	16	10	5	0.35
Nn50d6h6	6	1074	940	134	13	11	3	0.21
average	4	492	410	82	11	9	2	0.18

Table 6: Inter-cluster route incidence with $H = 3, 6$

The results show that the impact of the inter–cluster routes in the final solution is significant. In fact, the average value of the ratio for the data set with $H = 3$ is equal to 16%, while with $H = 6$ it is around 18%. Considering all the values reported in the last column of Table 6, it is possible to notice that the importance of inter–cluster routes increases with the size of the instances, reaching values that are higher than 50%. Therefore, the solution is very influenced by these routes. The potential of the matheuristic in the multi–depot case is to find a good trade–off between simplification of the problem (phase 1) and global optimization (phases 2 + 3). Therefore, solving a single–depot *IRP* for each cluster could be less efficient.

7. An application to the last-mile delivery: exploration of innovative distribution in a large city setting

In this section we present computational results related to a real case study. In particular, the impact of a *VMI* system applied to nanostores on a South East Asian city is analysed. The analysis is developed in collaboration with a supply chain team in Procter & Gamble (P&G), that provided the data set.

7.1. Instances description

P&G is one of the largest consumer goods companies and serves many different customers and consumers around the globe. A substantial part of its products is sold in nanostores. These traditional channel stores are independent, family-owned stores (such as kiosks, neighborhood stores, grocery and convenience stores) and they are typically a large share of the retail for emerging markets. **Nanostores** are characterized by high levels of fragmentation and small order quantities. This case study was conducted in a large metropolitan area in South East Asia (called the City). The data set is composed of a central depot and **947** nanostores to be replenished, characterized by different features: geographical coordinates, addresses, delivery volumes, actual scheduled replenishment days, orders quantities, for a total horizon of one month. In the current situation the city is divided into 12 ***zones***. Each zone is made up of base units, called ***areas***. Deliveries and truck assignment are organized for each area, separately. The total number of areas is 49. This particular distribution structure is mainly due to the mechanism used for ordering and selling products. Generally, one or more areas are committed to a salesman, that has the role to collect all the customers' orders from his areas and to organize the vehicle routes for the delivery. The information flow between the vendor and the customers uses very limited ICT systems and optimizations, therefore we wanted to explore the possibility of implementing a VMI system in resupplying these nanostores. The analysis has different aims:

1. testing and evaluating the performances of the proposed model and algorithm with real data provided by the company;
2. understanding the real value for the business in applying the VMI setting instead of the classical Customer Managed Inventory in this context, in order to consider the possibility of investments for building an information and communication technology supporting the VMI setting;
3. comparing two different scenarios of the supply chain structure: the single-depot and the multi-depot scenario.

A series of assumptions are considered: the capacity of the central depot is not binding; the fleet is homogeneous with the capacity of each vehicle equal to 300 boxes; the maximum driving time is eight hours per day; the maximum inventory level of each customer is set equal to 3/2 of the corresponding average daily demand, while the initial inventory is equal to 1/2 of its average daily demand. Moreover, in order to evaluate the advantages of the multi–depot VMI setting, the company identified some locations in which a secondary facility/depot could be installed to serve all the customers in each area. Since these facilities are already organized from the logistics point of view to operate as secondary depots, no additional cost is charged in the objective functions of our mathematical formulations.

7.2. Computational results

We use the same algorithm and infrastructure described before. The performance of the algorithm is evaluated on a set of 12 instances (one for each zone) with a number of customers ranging from 28 to 138. The number of depots increases from 3 to 11 according to the size of the instance. The time horizon H is 6 days. The following parameters are set: $\epsilon = 0.2$, $\omega = 10$, $\alpha = 0.2$, $TC = 6$, $CC \in \{3, \ldots, 23\}$ and $\lambda \in \{0.2, \ldots, 0.5\}$. A time limit of 3 hours was imposed for running the matheuristic.

7.2.1. Clustering results

In the first step of the algorithm we re–define the clusters for each zone, trying to create clusters that are balanced from the cardinality and the critical level points of view. Table 6 provides results and comparison between the original system and the optimized one. It is organized as follows. Columns Zone and N.Customers described the instance features. More specifically, column **Zone** provides the zone name, column **# customers** the number of customers into each zone. Column **# depots** is a feature of the new system and shows the number of optimized depots operating in each zone. Finally, columns **# areas** and Areas report the number of the areas and their cardinalities; they are introduced both for the original system and for the optimized one into the table.

Table 7 shows the number of the different areas we obtained in each zone. If we compare the results with the original situation, it is evident that the new clusters are more balanced and homogeneous.

		Original System		Optimized system		
Zone	**# customers**	**# areas**	**Areas**	**# depots**	**# areas**	**Areas**
Zone1	32	4	3,1,26,2	3	3	14,12,6
Zone2	28	9	1,1,18,3,1,1,1,1,1	2	2	14,14
Zone3	49	6	5,6,1,2,28,7	6	5	10,12,10,12,5
Zone4	59	10	2,2,21,2,1,1,5,1,1,23	3	3	23,13,23
Zone5	57	7	1,7,1,42,1,1,4	3	3	19,19,19
Zone6	135	16	46,1,2,6,3,2,21,10,1,8,5,2,4,2,2,20	10	10	14,14,14,14,14,14,14,14,9,14
Zone7	135	21	9,1,4,22,40,10,1,9,10,1,8,1,6,1,2,2,1,1,1,2,3	11	11	13,15,14,1,15,4,15,15,15,13,15
Zone8	138	21	6,33,4,5,14,7,9,6,2,23,3,6,7,3,2,2,1,1,2,1,1	11	11	12,15,15,15,15,10,5,15,6,15,15
Zone9	80	1	80	6	6	6,15,15,15,15,14
Zone10	51	1	51	3	3	19,13,19
Zone11	131	1	131	9	9	15,15,15,15,15,15,11,15,15
Zone12	52	9	3,4,1,1,19,22,1,1	3	3	18,16,18

Table 7: Clustering Results

7.2.2. *Matheuristic Results*

Table 7 presents the matheuristic results. We consider two different configurations: the multi-depot configuration ($MD - C$) and the single-depot configuration ($SD - C$). For the $MD - C$ we simply solve the instances with the algorithm described in the previous sections. For the $SD - C$ we modify the total routing cost introducing the cost of the direct delivery from each secondary depot to the main one. We compare the $SD - C$ with the actual situation in the City, in terms of total kilometres covered for the deliveries. Table 7 describes the results. Column **Zone** introduces the name of each zone, column **# Routes** the total number of routes generated by our algorithm, column $MD - C$**(km)** the solution provided by our algorithm for the multi-depot case, in terms of total kilometres routed for delivering freight in each Zone, column **Time(s)** the total computational time for each instance, while columns **TimeIRP** and **GAP %** the computational time and the CplexGAP related to model (27)–(35), respectively, column $SD - C$**(km)** the solution for the single-depot case, column **Actual(km)** the total kilometres routed in the City with the Customer Managed Inventory approach, column GAP_{SA} **%** the gap between the total delivery cost of the solution obtained with the VMI setting and total delivery cost paid by the Company, while GAP_{SM} describes the gap the cost of the solution with only one depot and the cost of the solution in which several secondary facilities/depots are used.

These gaps are computed as:

$$GAP_{SA} = (\frac{(SD-C)-Actual(Km)}{Actual(Km)}) * 100$$

$$GAP_{SM} = (\frac{(SD-C)-(MD-C)}{(MD-C)}) * 100.$$

Zone	# Routes	MD-C.(km)	Time(s)	TimeIRP(s)	Gap%	SD-C(km)	Actual(km)	Gap_{SA}%	Gap_{SM}%
Zone1	778	346.48	64.82	3.58	0.00	368.38	193.73	90.25	-5.94
Zone2	968	57.20	27.61	8.05	0.00	126.41	140.19	-9.83	-54.94
Zone3	1011	49.51	208.3	7.67	0.00	90.74	203.75	-55.46	-45.43
Zone4	4531	169.34	683.98	60.72	0.00	267.99	226.04	18.55	-36.80
Zone5	3544	232.13	890	35.44	0.00	309.73	365.42	-15.24	-25.05
Zone6	4447	339.64	5759.21	39.65	0.00	491.66	710.32	-30.78	-30.91
Zone7	4735	345.81	4603.71	37.21	0.00	536.52	710.28	-24.46	-35.54
Zone8	4574	349.70	3719.53	35.46	0.00	536.06	757.38	-29.22	-34.76
Zone9	2887	142.69	3057.89	18.56	0.00	196.69	344.50	-42.90	-27.45
Zone10	2747	115.56	121.5	7.73	0.00	145.65	232.17	-37.26	-20.65
Zone11	4983	71.34	6000	43.7	0.00	149.09	540.46	-72.41	-52.14
Zone12	2721	92.45	187.74	20.17	0.00	136.94	279.90	-51.07	-33.49
AVERAGE	3161		2110.36	26.49	0.00			-21.65	-33.05

Table 8: Matheuristic results

In general, it is possible to observe that all the instances are optimality solved in a short computational time. Columns 4 and 5 underline that most of the computational time is consumed in the routes generation phase, because the final optimization needs only few seconds for all instances. If we compare the case study results with the benchmark instances run with $H = 6$, we can underline that the average computational time is similar to the benchmark. Indeed, in the case study it is only 7.37% more than in the previous tests, while the average number of routes generated is more than 5 times with respect to the benchmark case. The increase in the number of routes is related to the particular structure of the network: customers are close each other and require small and frequent orders. The route generation phase is affected by both these factors, because it is driven by the cluster composition, by the borderline customers and by the capacity of the vehicle. In spite of this aspect, the total computational time does not increase so much. Column GAP_{SA} shows that the results for $SD-C$ are 21.65% better in terms of total kilometres reduction. In general, the algorithm is able to find a solution better than the current solution for about 92% of the instances. This is a valuable result, because it underlines the potential benefit of applying a *VMI* system also in the last part of the supply chain, where deliveries are fragmented and not easily optimized. These results underline that using a configuration with secondary facilities/depots can allow a further saving of about 33% in terms of kilometres. This aspect is a good point to investigate the benefits of the multi–depot *IRP* in emerging markets.

8. Conclusion

In this study, the *MDIRP* with a homogeneous fleet of vehicles is studied. While classical *IRPs* have been studied extensively, the multi-depot case represents a variant not well investigated despite its applications in last-mile delivery optimization, when the Vendor–Managed Inventory paradigm is applied. A MILP formulation is presented. An effective matheuristic algorithm to solve the *MDIRP* is designed and implemented. A take away message from this approach is that a good clustering and the corresponding intra-cluster routes are not enough to guarantee a good performance of the matheuristic. We showed that, in our approach, the main role is played by the inter-cluster routes, based on "borderline customers" among clusters. Furthermore, the clustering phase impacts also on the computational time of the matheuristic: if clusters are not so balanced in terms of critical level and cardinality, the computational time for generating the routes increases and slows down the solving process. We can conclude that a good clustering phase is the one able to produce, in short computational time, clusters that are a good basis to build effective intra-cluster routes. Our matheuristic clearly outperformed a branch–and–cut algorithm with several families of cuts. It was tested also on real instances related to the last mile deliveries to a large set of nanostores in Asia, where it shows to be effective and able to significantly improve the original solution.

References

References

[1] Barratt, M., Oliveira, A.. Exploring the experiences of collaborative planning initiatives. International Journal of Physical Distribution and Logistics Management 2001;31(4):266–289.

[2] Dauzère-Pérès, S., Nordli, A., Olstad, A., Haugen, K., Koester, U., Per Olav, M., et al. Omya Hustadmamor optimizes its supply chain for delivering calcium carbonate slurry to european paper manufacturers. Interfaces 2007;37(1):39–51.

[3] Bell, W., Dalberto, L., Fisher, M., Greenfield, A., Jaikumar, R., Kedia, P., et al. Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. Interfaces 1983;13:4–23.

[4] Grønhaug, R., Christiansen, M., Desaulniers, G., Desrosiers, J.. A branch-and-price method for a liquefied natural gas inventory routing problem. Transportation Science 2010;44(3):400–415.

[5] Shen, Q., Chu, F., Chen, H.. A lagrangian relaxation approach for a multi-mode inventory routing problem with transshipment in crude oil transportation. Computers & Chemical Engineering 2011;35(10):2113–2123.

[6] Popović, D., Vidović, M., Radivojević, G.. Variable neighborhood search heuristic for the inventory routing problem in fuel delivery. Expert Systems with Applications 2012;39(18):13390–13398.

[7] Christiansen, M., Fagerholt, K., Flatberg, T., Haugen, Ø., Kloster, O., Lund, E.H.. Maritime inventory routing with multiple products: A case study from the cement industry. European Journal of Operational Research 2011;208(1):86–94.

[8] Andersson, H., Hoff, A., Christiansen, M., Hasle, G., Løkketangen, A.. Industrial aspects and literature survey: Combined inventory management and routing. Computers & Operations Research 2010;37(9):1515–1536.

[9] Bertazzi, L., Savelsbergh, M., Speranza, M.. Inventory routing. In: The vehicle routing problem: latest advances and new challenges. Springer; 2008, p. 49–72.

[10] Campbell, A., Salvesbergh, M.. A decomposition approach for the inventory routing problem. Transportation Science 2004;38(4):488–502.

[11] Bertazzi, L., Speranza, M.. Inventory routing problems: an introduction. EURO Journal on Transportation and Logistics 2012;1:307–326.

[12] Bertazzi, L., Speranza, M.. Inventory routing with multiple customers. EURO Journal on Transportation and Logistics 2013;2:255–275.

[13] Coelho, L., Cordeau, J.F., Laporte, G.. Thirty years of inventory–routing. Transportation Science 2013;48:1–19.

[14] Archetti, C., Bertazzi, L., Laporte, G., Speranza, M.. A branch-and-cut algorithm for a vendor-managed inventory-routing problem. Transportation Science 2007;41(3):382–391.

[15] Solyalı, O., Süral, H.. A single supplier–single retailer system with an order-up-to level inventory policy. Operations Research Letters 2008;36(5):543–546.

[16] Coelho, L., Laporte, G.. Improved solutions for inventory-routing problems through valid inequalities and input ordering. International Journal of Production Economics 2014;155:391–397.

[17] Avella, P., Boccia, M., Wolsey, L.. Single-item reformulations for a vendor managed inventory routing problem: Computational experience with benchmark instances. Networks 2015;65(2):129–138.

[18] Avella, P., Boccia, M., Wolsey, L.. Single-period cutting planes for inventory routing problems. Transportation Science 2018;52(3):497–508.

[19] Archetti, C., Bertazzi, L., Hertz, A., Speranza, M.. A hybrid heuristic for an inventory routing problem. INFORMS Journal on Computing 2012;24(1):101–116.

[20] Bertazzi, L., Speranza, M.. Matheuristics for inventory routing problems. Hybrid algorithms for service, computing and manufacturing systems: routing and scheduling solutions IGI Global, Hershey 2011;:1–14.

[21] Aksen, D., Kaya, O., Salman, F., Akça, Y.. Selective and periodic inventory routing problem for waste vegetable oil collection. Optimization Letters 2012;6(6):1063–1080.

[22] Ekici, A., Özener, O.O., Gültekin, K.. Cyclic delivery schedules for an inventory routing problem. Transportation Science 2014;.

[23] Raa, B., Aghezzaf, E.H.. Designing distribution patterns for long-term inventory routing with constant demand rates. International Journal of Production Economics 2008;112(1):255–263.

[24] Aghezzaf, E.H., Zhong, Y., Raa, B., Mateo, M.. Analysis of the single-vehicle cyclic inventory routing problem. International Journal of Systems Science 2012;43(11):2040–2049.

[25] Vansteenwegen, P., Mateo, M.. An iterated local search algorithm for the single-vehicle cyclic inventory routing problem. European Journal of Operational Research 2014;237(3):802–813.

[26] Raa, B.. Fleet optimization for cyclic inventory routing problems. International Journal of Production Economics 2015;160:172–181.

[27] Coelho, L., Cordeau, J.F., Laporte, G.. Consistency in multi-vehicle inventory-routing. Transportation Research Part C: Emerging Technologies 2012;24:270–287.

[28] Adulyasak, Y., Cordeau, J.F., Jans, R.. Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems. INFORMS Journal of Computing 2013;14:103–120.

[29] Coelho, L., Laporte, G.. A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. International Journal of Production Research 2013;51(23-24):7156–7169.

[30] Cordeau, J.F., Laganà, D., Musmanno, R., Vocaturo, F.. A decomposition-based heuristic for the multiple-product inventory-routing problem. Computers & Operations Research 2015;55:153–166.

[31] Desaulniers, G., Rakke, J.G., Coelho, L.C.. A branch-price-and-cut algorithm for the inventory-routing problem. Transportation Science 2015;50(3):1060–1076.

[32] Archetti, C., Boland, N., Speranza, M.. A matheuristic for the multivehicle inventory routing problem. INFORMS Journal on Computing 2017;29(3):377–387.

[33] Kleywegt, A., Nori, V., Savelsbergh, M.. The stochastic inventory routing problem with direct deliveries. Transportation Science 2002;36(1):94–118.

[34] Kleywegt, A., Nori, V., Savelsbergh, M.. Dynamic programming approximations for a stochastic inventory routing problem. Transportation Science 2004;38(1):42–70.

[35] Solyalı, O., Cordeau, J.F., Laporte, G.. Robust inventory routing under demand uncertainty. Transportation Science 2012;46(3):327–340.

[36] Bertazzi, L., Bosco, A., Guerriero, F., Laganà, D.. A stochastic inventory routing problem with stock-out. Transportation Research Part C: Emerging Technologies 2013;27:89–107.

[37] Coelho, L., Cordeau, J.F., Laporte, G.. Heuristics for dynamic and stochastic inventory-routing. Computers & Operations Research 2014;52:55–67.

[38] Bertazzi, L., Bosco, A., Laganà, D.. Min–max exact and heuristic policies for a two-echelon supply chain with inventory and transportation procurement decisions. Transportation Research Part E: Logistics and Transportation Review 2016;93:57–70.

[39] Roldan, R., Basagoiti, R., Coelho, L.. A survey on the inventory-routing problem with stochastic lead times and demands. Journal of Applied Logic 2017;24:15–24.

[40] Bertazzi, L., De Maio, A., Laganà, D.. The impact of a clustering approach on solving the multi-depot IRP. In: International Conference on Optimization and Decision Science. Springer; 2017, p. 507–515.

[41] Bramel, J., Simchi-Levi, D.. A location based heuristic for general routing problems. Operations Research 1995;43(4):649–660.

[42] Ahr, D.. Contributions to multiple postmen problems, phd dissertation. University of Heidelberg 2004;:117.

[43] Fischetti, M., González, J.S., Toth, P.. Solving the orienteering problem through branch-and-cut. INFORMS Journal on Computing 1998;10(2):133–148.

[44] Gendreau, M., Laporte, G., Semet, F.. A branch-and-cut algorithm for the undirected selective traveling salesman problem. Networks 1998;32(4):263–273.

[45] Barahona, F., Grötschel, M.. On the cycle polytope of a binary matroid. Journal of Combinatorial Theory, Series B 1986;40(1):40–62.

[46] Padberg, M., Rinaldi, G.. Facet identification for the symmetric traveling salesman polytope. Mathematical Programming 1990;47(1-3):219–257.

[47] Aráoz, J., Fernández, E., Meza, O.. Solving the prize-collecting rural postman problem. European Journal of Operational Research 2009;196(3):886–896.

[48] Archetti, C., Bianchessi, N., Irnich, S., Speranza, M.. Formulations for an inventory routing problem. International Transactions in Operational Research 2014;21(3):353–374.

[49] Salhi, S., Imran, A., Wassan, N.. The multi-depot vehicle routing problem with heterogeneous vehicle fleet: Formulation and a variable neighborhood search implementation. Computers & Operations Research 2014;52:315–325.