UNIVERSITÀ DELLA CALABRIA

Department of Mathematics and Computer Science
PhD course on Logistics and Operations Research-
MAT/09 - Cycle XXX

PhD Thesis

Vendor Managed Inventory in the last-mile delivery: an application to an industry case and future challenges

Supervisor:
Prof. Demetrio Laganà

Coordinator:
Prof. Nicola Leone

Candidate:
Ing. Annarita De Maio

Academic Year 2016 - 2017

To my mother, to my father, to my sister,
for having taught me perseverance in reaching my goals
and for always supporting and accompanying me
during the journey that led me to this goal.

A mia madre, a mio padre, a mia sorella,
per avermi insegnato la perseveranza nel raggiungere i propri obiettivi
e per avermi sempre sostenuta ed accompagnata
durante il cammino che mi ha condotto a questo traguardo.

Contents

List of Figures

List of Tables

Introduction

In the last decades it was registered a significant move from towns to big cities. It is expected that the tendency will grow in the next future. This fact generates lots of impacts in the life of the cities. Firstly, an exponential increase in the consumption of different types of energies (water, electric energy, fossil fuel) is registered, secondly, a big growth of passengers and freight transport demand is generated. This situation produces high traffic and congestion level into dense urban areas, as a consequence. For the industrial point of view, it is crucial to have efficient delivery systems to reduce costs and transportation time, as long as to increase the customer satisfaction, especially in the last mile. Freight distribution efficiency, usually expressed in terms of customers' ordering timing, is an important issue for companies. For this reason, lots of companies use a Vendor Managed Inventory paradigm in their business, that has become a popular strategy for reducing inventory holding and distribution costs.

The thesis is part of a collaboration project with a big international company (from this point we refer to it as *FMCG Company* or simply as *Company*).

FMCG Company is an American multinational consumer goods corporation, operating in a wide range of cleaning and personal care products and selling them in approximately 180 countries. The distribution channels are different for size and typology, so they are represented by a large variety of retailers: grocery stores, drug stores, hyper and super markets, distributors, baby stores, e-commerce, high-frequency stores, pharmacies and nanostores.

The Company operates in different territories divided in emerging markets (Asia, Africa and Latin-America) and developed markets (North America and Western Europe). With the aim to improve the business while guarantee a widespread coverage of the deliveries, the Company faces very often lots of challenges related to the distribution in the last-mile delivery, especially into big and Mega cities [1].

For all these reasons, *FMCG Company* is interested in the potentiality of the application of a VMI system in the last-mile delivery, especially in a complex urban environment like a big/mega city, affected by the congestion and the presence of a large number of so-called "Nanostores", that are small stores featured by high frequency replenishment and small order sizes. For the Company, they are one of the big sectors for potential growth together with e-commerce, representing now around up to 40% of the business in relevant markets, especially the emerging-ones.

The research work has the aim to match the Company's desires, investigating the potential value of re-applying the VMI concept in the Nanostore channel into a big–city context and understanding the value that the system can generate in terms of transportation costs, comparing with the traditional current set-up, the Customer Managed Inventory.

In general, the VMI setting consists of a vendor/supplier or a manufacturer, and a set of customers/retailers located in a given geographical region. In the VMI system the supplier (vendor) monitors the inventory and decides the replenishment policy of each retailer (customers). The VMI assigns to the supplier the role of leading actor in the decisional process. More precisely, the supplier monitors retailers' inventory, checks inventory replenishment, and decides the delivery scheduling and routes. The main advantages offered by a VMI setting consist of costs' reduction due to the fact that the inventory management is performed by only the supplier instead of all the customers, and

[1] Mega-cities have a total population in excess more than 10 million inhabitants.

of improving the network workload due to the orders' fulfilments coordination. The use of the VMI paradigm usually allows to reach a competitive advantage for the Company in a long term planning horizon. Indeed, the company achieves great savings from:

- using tailored inventory policies to resupply the set of customers in each district as best as possible;
- optimizing of the deliveries to the stores, generating a decrease of the transportations cost;
- gaining the customer loyalty, by increasing the sense of customer satisfaction due to the reductions of delays and errors in delivering;
- reducing some negative effects on the supply chain like uncertainty, bullwhip effects and stock-outs;
- improving the flexibility in production;
- reducing inventory costs.

The aim of reducing the transportation cost, while making under control the inventory cost, needs the definition of Inventory Routing Problems ($IRPs$). $IRPs$ are combinatorial problems arising in the context of the VMI system, in which inventory control and routing decisions have to be made simultaneously. In particular, the vendor decides on the best replenishment policy of the customers by avoiding stock-out, respecting their maximum inventory capacity and achieving the minimization of the total transportation and inventory costs. As regard to the classical vehicle routing problems, the $IRPs$ show a growing complexity due to the integration of the inventory component into the decisional process devoted to define the best sequence in which customers must be serviced. The field is largely investigated in the past, a literature overview could be found in [17] and [25], while the successful industrial applications are described in [2].

The main idea for our research is to divide the big urban space into little districts to manage the distribution problem. In this way, it is possible to embed the VMI setting into the city logistics context. Each district presents a hub-shop, in a central position, that receives and storages freight for the delivers to the final customers. The hub-shop have to cover the demand of the entire district. We mainly study a variant of the classical problem that is the multi-depot IRP ($MDIRP$). As showed, the inventory routing problem has received a lot of attention over the last years. However, to the best of our knowledge, the multi-depot case was not so large investigated in the inventory routing literature. Indeed, the main contributions devoted to the MDIRP are highlighted in the following: a clustering technique to generate multi-depot instances was proposed in [45], while in [49] a mixed integer linear program for the multi-depot IRP is proposed. Lots of applications to the multi-depot case can be found following Vehicle Routing Problems papers, like [36] and [50], that represent a good inspiration for the research.

In the following we list the major contributions of this thesis:

- in the first part of the thesis the $MDIRP$ is formulated and two chapters are dedicated to its investigation. In *Chapter 1*, a simple *Branch-and-cut algorithm* is designed to solve the problem, and some valid inequalities are introduced. To reduce the complexity of the problem, a clustering method inspired by the *Capacitated Concetrator Location Problem* is used to pre–assign customers to the depots. This method allows to simplify the problem that takes place in a large urban space, while obtaining good solutions within a computational time smaller than the one required by the branch–and–cut algorithm. Indeed, the branch-and-cut algorithm combined with the clustering is faster than the simple version, while the resolution method remains very slow in general. The aim of this work is to show that, even if a clustering approach in a multi-depot problem is useful to improve the solution obtained in a given time limit, it is not sufficient

for finding high quality solutions. For these reasons, the focus of the next chapter consists of designing and of comparing different clustering approaches for the $MDIRP$. More specifically, a faster clustering model is developed, able to take into account several features that are typical of the IRP, and a complex heuristic approach is designed in order to solve large instances;

- in the *Chapter 2*, a complex *Matheuristic* is designed for solving the problem, it is divided in three main steps: a clustering optimization phase (in order to group the customers around each depot), a route generation phase (in order to build a set of feasible routes), a final optimization phase (in which a simplified version of the model is solved on the basis of feasible paths). Finally, the matheuristic is compared with the branch–and–cut algorithm, getting better solution quality within a computational time shorter than the one of the exact algorithm. This evidence allows us to propose to the Industry Company to test this algorithm on real case benchmarks;

- in the *Chapter 3*, a more complex version of the $MDIRP$ is designed. We added the possibility to deliver multi–product by using an heterogeneous fleet of vehicles. In this way, we modified the problem to make it as close as possible to the real scenario in which the Company works. In this case, a *Variable MIP Neighborhood Search algorithm* is designed to solve the problem. We present computational results for some classical benchmark and ad hoc data set instances;

- in the *Chapter 4*, an application of the matheuristic for the $MDIRP$ using real data for Surabaya city (Isle of Jackarta) is presented. The aim of this Chapter is to show the effectiveness of the VMI setting applied to the last–mile delivery of the Nanostores in a real context. We are able to prove that the VMI has a good impact on the business of the Company,

because it is able to reduce a big portion of the transportation costs that the Company actual pays to ship freights to the stores in a wide area.

Chapter 1

The impact of a clustering approach on solving the Multi-Depot IRP

Joint work with *Luca Bertazzi* and *Demetrio Laganà*
Published on: Optimization and Decision Science: Methodologies and Applications. ODS 2017. Springer Proceedings in Mathematics & Statistics, vol 217. Springer, Cham

Abstract: We study the *Multi-Depot Inventory Routing Problem* (*MDIRP*) with homogeneous vehicle fleet and deterministic demand. We implement a branch-and-cut algorithm for this problem. Then, we design a matheuristic in which we first optimally solve a modified version of the *Capacitated Concentrator Location Problem* (*CCLP*) to generate a cluster of customers for each depot and, then, we exactly solve the problem based on these clusters with a branch-and-cut algorithm. Computational results are presented to compare the performance of the matheuristic with respect to the branch-and-cut, in order to analyze the value of the clustering approach in solving this problem.

Keywords: Inventory routing, branch-and-cut, clustering.

1.1 Introduction

Inventory Routing Problems (*IRPs*) spread out in the integrated optimization of inventory and distribution management in supply chains. This is a win-win approach in which a supplier coordinates the replenishment of a set of customers, deciding when to visit each customer over a time horizon, the quantities to deliver and the routes to travel. These problems received a remarkable attention in the recent decades. Examples of real industrial applications can be found in the survey presented by [2]. Different versions of the problem were investigated in the literature, for example the single and multi-vehicle case, the single and multi-product case, the cases with deterministic, stochastic and robust demand. For an in depth analysis of the state of the art, the reader can refer to the tutorials by [17] and [18] and to the survey by [25].

IRPs are well known to be NP-hard problems. For this reason, the main challenge is to design efficient exact algorithms on one side and effective heuristic algorithms on the other side. Different solution methods were proposed in the past for the single-depot case: exact methods can be found in [3] and [26], while decomposition approaches are proposed by [22] and [29]. The Multi-Depot *IRP* ($MDIRP$) is not largely investigated in literature. A clustering technique to generate multi-depot instances was proposed by [45], while a formulation for a multi-depot and multi-commodity *IRP* was introduced by [49]. To the best of our knowledge, no branch-and-cut algorithms are available for it. Instead, the multi-depot case was studied for the simpler *VRPs*, where the clustering of customers for each depot is often used. Effective clustering techniques are designed also for Location–Routing Problems. The most recent and complete survey about this class of problems is presented by [48] and an effective application can be found in [20].

Our contribution is to provide the first branch-and-cut algorithm for the *MDIRP* and to design a matheuristic algorithm, based on an optimal solution of a variant of the classical *Capacitated Concentrator Location Problem* (*CCLP*). Computational results are presented to evaluate the impact of the clustering on the solution quality and on the computational time, with respect to the best solution provided by the branch-and-cut.

The remainder of the paper is organized as follows. The $MDIRP$ is formally described and formulated in Section 1.2. The branch-and-cut algorithm is described in Section 1.3. The matheuristic we propose is described in Section 1.4. The computational results are shown in Section 1.5. Finally, conclusions are drawn in Section 1.6.

1.2 Problem Description and Formulation

In this section, the *MDIRP* is described and the corresponding mathematical formulation based on binary edge-variables is presented. This formulation is an extension of the single–vehicle and single–depot *IRP* formulation proposed by [3]. We consider a complete undirected graph $G(V, E)$. A set of depots $P = \{1, 2, \ldots, l\}$ delivers a product to a set $I = \{1, 2, \ldots, n\}$ of customers. The set with all vertices is denoted by $V = P \cup I$. The parameter c_{ij} is the travelling cost of the edge $(i, j) \in E$. These costs satisfy the triangle inequality. Given S (a proper and non–empty subset of vertices), $S \in I$, $E(S)$ denotes the set of edges (i, j), such that $i, j \in I$. The set of the vehicles is denoted by $K = \{1, 2, \ldots, M\}$. Each vehicle has capacity C. A discrete time horizon H is given. The set of time periods is denoted by $T = \{1, 2, \ldots, H\}$. Each customer $i \in I$ defines a maximum inventory level U_i and has a given starting inventory level $Inv_{i0} \leq U_i$. At each time $t \in T$, each customer i has to satisfy the deterministic demand d_{it}. In this problem split delivery is allowed. More specifically, two types of split may occur: the first split delivery consists of

servicing the same customer with two vehicle–routes departing from the same depot in the same time period, while the second split delivery occurs when the same customer is visited by two vehicle–routes departing from different depots in the same time period.

The variable Inv_{it} indicates the inventory level of customer i at the begin of period t. The time $H+1$ is included in the inventory computation to take into account the consequences of the decisions at time H. The variable y_{iktp} represents the quantity delivered to customer i in period t by vehicle k from depot p. The binary variable x_{ijktp} is equal to 1 if the edge (i,j) is traversed in period t by vehicle k starting from depot p. The binary variable z_{iktp} is equal to 1 if customer i is visited in period t by vehicle k from depot p. The binary variable z_{pktp} is equal to 1 if vehicle k located in depot p starts its tour from depot p in period t. This problem can be formulated as follows:

$$Min \sum_{t\in T}\sum_{i\in V}\sum_{j\in V}\sum_{k\in K}\sum_{p\in P} c_{ij}x_{ijktp} \tag{1.1}$$

s.to.

$$Inv_{i,t+1} = Inv_{it} + \sum_{k\in K}\sum_{p\in P} y_{iktp} - d_{it} \qquad t\in T,\ i\in I \tag{1.2}$$

$$\sum_{p\in P}\sum_{k\in K} y_{iktp} + Inv_{it} \leq U_i \qquad t\in T,\ i\in I \tag{1.3}$$

$$\sum_{i\in I} y_{iktp} \leq C\ z_{pktp} \qquad t\in T,\ p\in P,\ k\in K \tag{1.4}$$

$$y_{iktp} \leq C\ z_{iktp} \qquad i\in I,\ t\in T,\ p\in P,\ k\in K \tag{1.5}$$

$$\sum_{i\in I} y_{iktp} \geq z_{pktp} \qquad t\in T,\ p\in P,\ k\in K \tag{1.6}$$

$$z_{bktp} = 0 \qquad \forall\ t\in T,\ p\in P,\ k\in K,\ b\in P(p\neq b) \tag{1.7}$$

$$\sum_{p\in P} z_{pktp} \leq 1 \qquad t \in T,\ k \in K \tag{1.8}$$

$$\sum_{p\in P} z_{iktp} \leq 1 \qquad t \in T,\ k \in K,\ i \in I \tag{1.9}$$

$$\sum_{j\in I, j<i} x_{ijktp} + \sum_{j\in I, j>i} x_{ijktp} = 2\ z_{iktp} \qquad i \in I,\ t \in T,\ p \in P,\ k \in K \tag{1.10}$$

$$\sum_{(i,j)\in E(S)} x_{ijktp} \leq \sum_{i\in S} z_{iktp} - z_{uktp} \qquad S \subseteq I, |S| \geq 2, t \in T, \quad k \in K, p \in P, \text{ for a given } u \in S \tag{1.11}$$

$$x_{ijktp} \in \{0,1\} \qquad i,j \in V,\ t \in T,\ p \in P,\ k \in K \tag{1.12}$$

$$Inv_{it} \geq 0 \qquad i \in I,\ t \in T \tag{1.13}$$

$$y_{iktp} \geq 0 \qquad i \in I,\ t \in T,\ p \in P,\ k \in K \tag{1.14}$$

$$z_{iktp} \in \{0,1\} \qquad i \in I,\ t \in T,\ p \in P,\ k \in K. \tag{1.15}$$

The objective function (1.1) states the minimization of the total routing cost. Constraints (1.2)-(1.3) are inventory constraints at the customers. Constraints (1.4)-(1.5) are capacity constraints. Constraints (1.6)-(1.9) define the multi-depot case and split delivery. Constraints (1.10)-(1.11) are classical routing constraints. Constraints (1.12)-(1.15) define integrality and non-negativity variables conditions.

1.3 A Branch-and-Cut Algorithm

In order to exactly solve the *MDIRP* described in the previous section, we design and implement the following branch-and-cut algorithm. The *sub-tour elimination constraints* (1.11) were initially removed from the formulation (1.1)-(1.15) and added dynamically using the separation procedure described in [46]. They were introduced considering a given $u \in S$, for which the following condition is valid: $u = argmax_i\{\bar{z}_{iktp}\}$, where $\bar{z}_{iktp}$ is the value of variable z_{iktp} in the current LP relaxation. At each tree node, the violated (1.11) are found and added to the current sub-problem that is then optimized. If no violations are identified, branching occurs at the current node. No priority variables are defined for the branching strategy. In order to improve the quality of the root node lower bound of the branch-and cut tree, the following valid inequalities are added to the initial *LP*.

1. *Priority inequalities*:

$$z_{iktp} \leq z_{pktp} \qquad i \in I,\ t \in T,\ p \in P,\ k \in K. \tag{1.16}$$

   These valid inequalities are used by [3]. We consider an adapted version of them for the *MDIRP*.

2. *Logical inequalities*:

$$x_{ipktp} + x_{piktp} \leq 2\ z_{iktp} \qquad i \in I,\ t \in T,\ p \in P,\ k \in K. \tag{1.17}$$

$$x_{ijktp} \leq z_{iktp} \qquad i, j \in I,\ t \in T,\ p \in P,\ k \in K. \tag{1.18}$$

   These inequalities are inspired by the logical cuts by [37] and [38].

3. *Aggregate parity inequalities*:

$$\sum_{p\in P}\sum_{(i,j)\in\delta(S)} x_{ijktp} \geq \sum_{p\in P}\sum_{(i,j)\in F} x_{ijktp} - |F| + 1, \quad t \in T,\ k \in K, \\ F \subseteq \delta(S), |F| \text{ odd.} \tag{1.19}$$

4. *Disaggregate parity inequalities*:

$$\sum_{(i,j)\in\delta(S)} x_{ijktp} \geq \sum_{(i,j)\in F} x_{ijktp} - |F| + 1, \quad t \in T,\ p \in P,\ k \in K, \\ F \subseteq \delta(S), |F| \text{ odd.} \tag{1.20}$$

Parity inequalities are initially defined in [9] as co-circuit inequalities. They are really effective for problems with binary variables, in case the parity of vertices is required. Inequalities (1.19)-(1.20) are separated heuristically following the procedure described by [7].

1.4 A Matheuristic for the *MDIRP*

As explained before, solving *MDIRP* with an exact method is really complex. The branch-and-cut algorithm is very slow even in small instances. Therefore, we design a matheuristic algorithm based on the following three steps:

1. Optimally solve the *CCLP* described below to generate clusters composed by a depot and a subset of customers.

2. Import the clusters in the *MDIRP* model: for each customer i not associated with depot p the corresponding z_{iktp} variables are set to zero, in order to forbid routes serving customers not associated to p.

3. Apply the branch-and-cut described in Section 1.3 to the model obtained in step 2.

Let us now describe the *CCLP* we solve in the matheuristic. The *CCLP* was largely investigated in the literature for the *VRP*. Different formulations and variants are described by [40]. We present an adapted formulation of the classical *CCLP* in order to match with the *MDIRP* case. We define the set P of depots as the set of potential concentrators. Let γ be the number of concentrators to open, R_p be the fixed cost to make p as a concentrator, D_i be the demand of customer i used to build the clusters, Γ_{pi} be the cost to assign customer i to concentrator p, Φ_p be capacity of each concentrator. The model involves two sets of binary variables: g_{pi} equal to 1 if the customer i is assigned to depot p and b_p equal to 1 if p is selected to be a concentrator. The mathematical formulation is described below:

$$Min \sum_{p \in P} \sum_{i \in I} \Gamma_{pi} g_{pi} + \sum_{p \in P} R_p b_p \tag{1.21}$$

s.t.

$$\sum_{p \in P} g_{pi} = 1 \quad i \in I \tag{1.22}$$

$$\sum_{i \in I} D_i g_{pi} \leq \Phi_p b_p \quad p \in P \tag{1.23}$$

$$\sum_{p \in P} b_p = \gamma \tag{1.24}$$

$$b_p \in \{0, 1\} \quad p \in P \tag{1.25}$$

$$g_{pi} \in \{0, 1\} \quad p \in P, \quad i \in I. \tag{1.26}$$

The objective function (1.21) minimizes the total cost to build the clusters. Constraints (1.22) guarantee that each customer is assigned to exactly one cluster. Constraints (1.23) ensure that the capacity of each cluster is not violated. Constraint (1.24) guarantee that γ clusters are built. Constraints (1.25)-(1.26) impose that the variables are all binary.

Instance	Time	N.Clust.	Cardinal.	Instance	Time	N.Clust.	Cardinal.
n5d2h3	40ms	2	3–2	n5d2h3	30ms	2	3–2
n10d2h3	50ms	2	4–6	n10d2h3	40ms	2	4–6
n15d2h3	70ms	2	6–9	n15d2h3	35ms	2	8–7
n20d30h3	50ms	3	5–11–4	n2d30h3	70	3	4–12–4
n25d4h3	60ms	4	5–9–4–7	n25d4h3	50ms	4	9–5–4–7
n30d4h3	80ms	4	5–12–4–9	n30d4h3	70ms	4	5–11–5–9
n5d2h6	40ms	2	3–2	n5d2h6	30ms	2	3–2
n10d2h6	50ms	2	4–6	n10d2h6	55ms	2	3–7
n15d2h6	30ms	2	11–4	n15d2h6	3ms0	2	11–4
n20d3h6	90ms	3	4–9–7	n20d3h6	80ms	3	12–4–4

Table 1.1: CCLP computational results

1.5 Computational Results

The branch-and-cut and the matheuristic described in Section 1.3 and 1.4 were implemented in C++ by using IBM Concert Technology and CPLEX 12.6, and run on an Intel Core i7-6500U 2.50 GHz and 8 GB RAM personal computer. An adapted version of two subsets of instances derived from the benchmark instances provided in [3] for the single-depot IRP are used. Instances are labeled as *nNdDhH*, where N is the number of customers, D is the number of the depots, H is the time horizon. A time limit of 2 hours is set for both the branch-and-cut and the matheuristic.

Table 1 provides the results obtained by solving the *CCLP* with the following data: $\gamma = |P|$, $R_p = 0$, D_i equal to the average demand over the time horizon, $\Gamma_{pi} = c_{pi}$, $\Phi_p = MC$. For each instance, it gives the computational time (ms), the number of clusters and their cardinality. The table is organized as follows: column **Instance** describes the instance label, column **Time** reports the computational time, column **N. Clust.** describes the number of generated clusters and column **Cardinal.** reports the cardinality of each cluster.

The results show that the clustering phase is no time consuming and the cardinality of the clusters is enough homogeneous in each instance.

	Matheuristic						Branch-and-cut						GAP(%)
Instance	Time(s)	Soub.	Par.	Cost	LB	GAP	Time(s)	Soub.	Par.	Cost	LB	GAP	
n5d2h3	51.35	22	198	1148.8	1148.8	0.00	210.77	127	436	1148.8	1148.8	0.00	0.00
n10d2h3	1150.83	230	677	2214.29	2214.29	0.00	t.l.	1008	1295	2149.16	1972.55	8.22	3.00
n15d2h3	t.l.	554	925	4117.12	3783.77	8.10	t.l.	1785	1802	4614.66	3383.05	26.69	-11.00
n20d3h3	t.l.	572	992	3467.5	3205.57	7.55	t.l.	1604	1954	3629.04	2579	28.93	-4.00
n25d4h3	t.l	369	728	3917.78	3536.76	9.04	t.l	/	/	/	/	/	
n30d4h3	t.l.	603	1185	4401.29	4142.78	5.87	t.l	/	/	/	/	/	
n5d2h6	5.06	51	185	2705.04	2705.04	0.00	5003.34	414	1035	2595.14	2595.14	0.00	4.00
n10d2h6	t.l.	444	1245	4939.37	4270.29	13.55	t.l.	1295	1610	5117.57	2971.66	41.93	-3.00
n15d2h6	t.l.	1264	2811	10830.5	7601.98	29.81	t.l.	1725	3565	10850	6795.24	37.37	-0.10
n20d3h6	t.l.	723	2278	11242.9	8899.94	20.84	t.l.	/	/	/	/	/	
Instance	Time(s)	Soub.	Par.	Cost	LB	GAP	Time(s)	Soub.	Par.	Cost	LB	GAP	
n5d2h3	106.17	27	216	956.38	956.30	0.00	112.79	98	314	956.38	956.38	0.00	0.00
n10d2h3	t.l.	173	560	2572.43	2346.81	9.61	t.l.	718	1072	2653.39	1789.64	48.27	-3.05
n15d2h3	t.l.	325	649	2436.43	2202.38	10.62	t.l.	573	1080	2436.43	2154.64	13.07	0.00
n20d3h3	3048.27	336	837	3837.28	3837.28	0.00	t.l.	1022	2005	4969.69	3460.06	43.63	-22.78
n25d4h3	t.l	360	472	3917.78	3536.76	9.04	t.l	/	/	/	/	/	
n30d4h3	t.l.	493	784	4415.79	4118.8	7.21	t.l	/	/	/	/	/	
n5d2h6	t.l.	44	704	5243.49	5218.33	0.48	t.l.	384	932	6304.28	4289.37	46.97	-16.82
n10d2h6	t.l	458	991	5530.09	4926.71	12.24	t.l.	1141	1793	6055.99	3609.7	67.97	-8.68
n15d2h6	t.l.	864	1901	9047.82	6880.83	31.49	t.l.	/	/	/	/	/	
n20d3h6	t.l.	726	1512	6748.37	5884.97	12.97	t.l	/	/	/	/	/	

Table 1.2: Matheuristic vs. Branch-and-cut

Table 2 compares the results obtained by applying the branch-and-cut algorithm introduced in Section 1.3 and the matheuristic described in Section 1.4. For each instance, it shows, both for the matheuristic and the branch-and-cut, the computational time (seconds) or t.l. when the time limit is reached (column **Time(s)**), the total number of added inequalities (subtour elimination constraints (column **Soub.**), disaggregated and aggregated parity inequalities (column **Par.**)), the cost of the best feasible solution found in the time limit (column **Cost**), the CPLEX lower bound value (column **LB**) and the CPLEX GAP (column **GAP**). In the last column the percentage GAP between the feasible solutions of two approaches is shown.

The results show that the matheuristic is always able to find a feasible solution in the time limit, while the branch-and-cut is not able to find it in 35% of the instances. Moreover, the CPLEX GAP of the matheuristic is about 9% on average, while it is about 26% in the solved instances. Note that the CPLEX GAP increases in the instances with time horizon $H = 6$ with respect

to the instances with $H = 3$. The last column of Table 1.2 shows that the matheuristic is able to find better solutions than the branch-and-cut in the same time limit, with a maximum reduction of about 23% of the cost. This underlines that the matheuristic is more efficient in terms of solution quality and computational time.

1.6 Conclusion

We studied the *Multi-Depot Inventory Routing Problem* ($MDIRP$). This problem is very difficult to be solved to optimality. A branch-and-cut algorithm designed for it was able to find a feasible solution in only 65% of the instances and provides an average CPLEX GAP of 26% in the solved instances. Our results showed that embedding the clusters obtained by optimally solving a variant of the *Capacitated Concentrator Location Problem* in the branch-and-cut allowed us to always find a feasible solution of the problem, to reduce the CPLEX GAP to 9% and to find better solutions. Future research could be devoted to improve this matheuristic, trying to strongly enhance the clustering approach.

Chapter 2

A Matheuristic Algorithm for the Multi-Depot Inventory Routing Problem

Joint work with *Luca Bertazzi, Leandro C. Coelho, Demetrio Laganà*
Accepted by Transportation Research Part E

The *Vendor-Managed Inventory* paradigm has important applications in modern supply chain management. One of these applications is the *Multi-Depot Inventory Routing Problem* ($MDIRP$), where several depots serve the demand of a set of customers over time. The supplier has to determine how to serve the customers from different depots, managing the inventory levels at the customers to avoid stock-outs, with the aim of minimizing the routing cost. This is an NP-hard problem that aims at optimizing the trade-off between inventory and routing management 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 in significantly shorter computational time. The solutions provided by the matheuristic algorithm are compared with the ones obtained with a branch–and–cut algorithm. Computational experiments

on classical single–depot Inventory Routing Problem instances and on a new dataset of $MDIRP$ instances show that the proposed matheuristic is very effective.

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

2.1 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. The main contributions in this direction are devoted to develop optimization models focused on two or more sequential logistics activities in the supply chain, such as inventory and routing, production, inventory and routing, location and routing. All the resulting optimization models are based on two–echelon networks, in which one or more depots deliver products to many retailers or customers.

Our problem falls within the field of the two–echelon optimization problems in which customers are supplied from a set of depots over a finite planning horizon. In this setting, the *Vendor-Managed Inventory* (VMI) paradigm gained importance in different companies in the last years. In this paradigm, the supplier monitors the inventory levels of each retailer and decides a replenishment policy. The supplier has the role of leading actor in the decisional process, in order to establish when and how much to deliver, and the routes traveled by 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 development of the VMI setting derives from successful industrial applications in different fields: Procter & Gamble

and Walmart ([10]), chemical products ([32]), oil and gas ([12], [41], [51]), fuel ([47]) and maritime cement transportation ([30]). For an in depth overview of VMI applications we refer to ([2]).

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 the multi-period decisional process. Usually, an IRP deals with minimizing the sum of inventory and routing costs during the planning horizon, while avoiding stock–outs at the customers. Three main decisions are relevant in an IRP: when and how much to deliver to customers and which routes to use. Uncertain IRP has been not widely investigated due to the increased complexity of the problem (see [52], [18], [28], [13]). Despite the fact that supply chains are usually characterized by a great level of uncertainty, the literature demonstrates that good results also derive from deterministic analysis. Deterministic IRPs are common both from a practical standpoint or as a research area. The single–product and single–vehicle IRP is studied in [3], [53]. Several papers investigate the single–product and multi–vehicle IRP, including [27] and [1]. The multi–product and multi-vehicle IRP is studied in [23] and [29]. Exact algorithms for the IRP include branch–and–cut, branch–and–price, and more recently, branch–and–price–and–cut ([33]). Heuristic methods were successfully applied to large IRP instances. A basic heuristic approach consists of decomposing the IRP in sub–problems solved in hierarchical order. Recent years have seen a large use of hybrid heuristic algorithms, in which mathematical programming models are embedded into heuristic frameworks. These algorithms are referred to as *matheuristics* and provide good results in solving IRPs, as shown by [5] for the single-vehicle case and by [4] for the multi-vehicle case. We refer to [16] for an introduction to matheuristics in solving IRPs. A decomposition approach to tackle large scale instances of the IRP can be found in [22], and [29]. For an in depth analysis

of the state of the art, the reader can refer to the tutorials by [17] and [18] and to the survey by [25].

We study an IRP in which the deterministic demand of a set of customers is satisfied from a set of depots. 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, while the corresponding inventory cost is paid by the customers. Therefore, the supplier minimizes the routing cost only, but guarantees that no stock–out occurs at the customers. The aim is to determine the vehicles assigned to each depot on each day, the customers served by each depot on each day, when and how much to deliver to each customer, and the delivery routes to use on each day. 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. We formulate this problem as a mixed-integer linear programming model and we design a three–phase matheuristic to effectively solve the problem in much shorter computational time. In the first phase of the matheuristic, an integer program is used to build clusters of customers on the basis of a quantitative measure of a customer critical inventory level. The importance of driving the solution with a clustering approach was discussed in [19], where a clustering procedure based on the solution of the classical *Capacitated Concentrator Location Routing Problem* (see [20]) is embedded in a branch-and-cut. The cited work underlined that a clustering approach in a multi-depot problem is useful for obtaining good solutions, but that a fast

heuristic procedure to perform this step is necessary. This is the motivation for a fast clustering model in our approach, able to incorporate several problem features. In the second phase, intra and inter-cluster routes are generated by considering the limited amount of resources in terms of vehicle capacity and maximum number of customers that can be served from a supplier. In the third phase, a 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 by solving the mixed integer linear programming formulation of the problem with a branch–and–cut algorithm.

The remainder of the paper is organized as follows. The problem is formally described in Section 2.2. In Section 2.3 a mathematical formulation of the $MDIRP$ is presented. The branch–and–cut algorithm is described in Section 2.4. The matheuristic algorithm is presented in Section 2.5. In Section 2.6 extensive computational results are presented and discussed. Finally, conclusions are drawn in Section 2.7.

2.2 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 $P = \{1, 2, \ldots, m\}$ is the set of depots that deliver a product to the set of customers $I = \{m+1, m+2, \ldots, m+n\}$ over a finite and discrete time horizon H. Let $T = \{1, 2, \ldots, H\}$ be the corresponding set of time periods. 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 $K = \{1, 2, \ldots, M\}$ of vehicles, with the same 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 $i \in I$ can be served by different vehicles from different depots in the same time period $t \in T$. Two types of split delivery

are permitted: when on a given period customer is visited by two vehicles from the same or 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. In each time period t, a deterministic integer demand d_{it} of each customer i must be satisfied. 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.

2.3 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 [3], 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 time period t;
- y_{iktp}: quantity to deliver to customer i in time period t by vehicle k starting the route 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 time period t, and equal to 0 otherwise (these variables are defined for each $i < j$);
- z_{iktp}: binary variable equal to 1 if vehicle k from depot p visits customer i in time period t, and equal to 0 otherwise;
- z_{pktp}: binary variable equal to 1 if vehicle k, assigned to depot p, starts its route from depot p in time period t, and equal to 0 otherwise.

The mathematical formulation is described by (2.1)–(2.14).

$$\min \quad \sum_{t\in T}\sum_{(i,j)\in E}\sum_{k\in K}\sum_{p\in P} c_{ij}x_{ijktp} \tag{2.1}$$

$$\text{s.t.} \quad I_{it} = I_{it-1} + \sum_{k\in K}\sum_{p\in P} y_{ikt-1p} - d_{it-1} \qquad \forall t \in T \cup \{H+1\}, \forall i \in I \tag{2.2}$$

$$I_{it} + \sum_{p\in P}\sum_{k\in K} y_{iktp} \leq U_i \qquad \forall t \in T, \forall i \in I \tag{2.3}$$

$$\sum_{i\in I} y_{iktp} \leq C\ z_{pktp} \qquad \forall t \in T, \forall p \in P, \forall k \in K \tag{2.4}$$

$$\sum_{i\in I} y_{iktp} \geq z_{pktp} \qquad \forall t \in T, \forall p \in P, \forall k \in K \tag{2.5}$$

$$y_{iktp} \leq C\ z_{iktp} \qquad \forall i \in I, \forall t \in T, \forall p \in P, \forall k \in K \tag{2.6}$$

$$z_{bktp} = 0 \qquad \forall t \in T, \forall p, b \in P, p \neq b, \forall k \in K, \tag{2.7}$$

$$\sum_{p \in P} z_{pktp} \leq 1 \qquad \forall t \in T, \forall k \in K \tag{2.8}$$

$$\sum_{j \in V:(j,i) \in E} x_{jiktp} + \sum_{j \in V:(i,j) \in E} x_{ijktp} = 2z_{iktp} \qquad \forall i \in V, \forall t \in T, \forall p \in P, \forall k \in K \tag{2.9}$$

$$\sum_{(i,j) \in E(S)} x_{ijktp} \leq \sum_{i \in S} z_{iktp} - z_{uktp} \qquad \forall S \subseteq I, |S| \geq 2, \forall t \in T, \forall k \in K, \forall p \in P, u \in S \tag{2.10}$$

$$x_{ijktp}, x_{pjktp}, x_{jpktp} \in \{0, 1\} \qquad \forall i, j \in I, \forall t \in T, \forall p \in P, \forall k \in K \tag{2.11}$$

$$I_{it} \geq 0 \qquad \forall i \in I, \forall t \in T \cup \{H + 1\} \tag{2.12}$$

$$y_{iktp} \geq 0 \qquad \forall i \in I, \forall t \in T, \forall p \in P, \forall k \in K \tag{2.13}$$

$$z_{iktp} \in \{0, 1\} \qquad \forall i \in V, \forall t \in T, \forall p \in P, \forall k \in K. \tag{2.14}$$

The objective function (2.1) minimizes the total routing cost. Constraints (2.2) balance the inventory flow at the customers. Note that the variable I_{i1} is equal to the initial inventory level I_{i0}. Constraints (2.3) guarantee that the maximum inventory level of each customer does not exceed U_i. Constraints (2.4) 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 (2.5) guarantee that, if the total quantity is greater than 0, then the corresponding vehicle is used. Constraints (2.6) 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 (2.7) guarantee that a vehicle k assigned to depot b in time period t cannot start a route from a different depot p in the same time period. Constraints (2.8) impose that each vehicle k can be assigned to at most one depot p in each time period t. Constraints (2.9) and (2.10) control the degree of the vertices and prohibit subtours, respectively. Finally, constraints (2.11)–(2.14) define the integrality and non-negativity conditions for the variables. Note that constraints (2.12) also guarantee that no stock-out occurs at each customer.

2.4 A Branch–and–cut algorithm

We design the following branch–and–cut algorithm to solve the problem. The goal is to solve the mathematical formulation (2.1)–(2.14) obtaining solutions with which to compare the performance of the matheuristic. The initial linear programming relaxation (LP) is obtained by removing constraints (2.10) from the problem formulation (2.1)–(2.14), adding the following constraints that guarantee that each customer i is visited at most once from vehicle k in each time period t:

$$\sum_{p \in P} z_{iktp} \leq 1 \quad \forall t \in T, \forall k \in K, \forall i \in I, \tag{2.15}$$

and then by adding violated subtour elimination constraints for each period, for each vehicle and for each depot. Constraints (2.15) are redundant in model (2.1)–(2.14), but they can increase the value of the corresponding LP. Subtour elimination cuts are separated heuristically along the lines of the procedure designed by [49]. 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 with the depot p is found, the corresponding subtour elimination constraint is added for $u = argmax_{i \in S_p} \{\overline{z}_{iktp}\}$, 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 the variables x and z, then constraints (2.10) can be violated only if a subset of customers is not connected with the depot p. Therefore, all violations of constraints (2.10) 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 find exactly 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:*

$$z_{iktp} \leq z_{pktp}, \quad \forall i \in I, \forall k \in K, \forall p \in P, \forall t \in T. \tag{2.16}$$

These valid inequalities were presented by [3] to improve the performance of the proposed branch-and-cut for the single–vehicle IRP.

2. *Logical inequalities:*

$$x_{ipktp} + x_{piktp} \leq 2\ z_{iktp}, \quad \forall i \in I, \forall t \in T, \forall p \in P, \forall k \in K \tag{2.17}$$

$$x_{ijktp} \leq z_{iktp}, \quad \forall i, j \in I, \forall t \in T, \forall p \in P, \forall k \in K. \tag{2.18}$$

These inequalities are inspired by the logical cuts of [37] and [38]. Inequalities (2.17) 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 (2.18) 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:*

$$\sum_{(i,j)\in\delta(S)\setminus(F)} x_{ijktp} \geq \sum_{(i,j)\in(F)} x_{ijktp} - |F| + 1,$$
$$\forall t \in T, \forall p \in P, \forall k \in K, \forall F \subseteq \delta(S), |F| \text{ odd} \tag{2.19}$$

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 p 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:*

$$\sum_{p \in P} \sum_{(i,j) \in \delta(S) \setminus (F)} x_{ijktp} \geq \sum_{p \in P} \sum_{(i,j) \in (F)} x_{ijktp} - |F| + 1,$$
$$\forall t \in T, \forall k \in K, \forall F \subseteq \delta(S), |F| \text{ odd} \tag{2.20}$$

These inequalities are added dynamically to the LP in each node of the branch–and–cut algorithm. Parity inequalities were introduced by [9] 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 [46] for the Symmetric TSP polytope. Inequalities (2.19) and (2.20) are separated heuristically according to the procedure described by [7].

2.4.1 Initial solution

An initial solution of the branch-and-cut is computed as follows.

We solve a relaxed formulation of (2.1)–(2.14), 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 (2.9) and (2.10) 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 (2.1) 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 $\overline{Z}_{ktp} = \{i \in I : \overline{z}_{iktp} = 1\}$ 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:

$$\sum_{i \in B_{ktp}} (1 - z_{iktp}) + \sum_{i \in I \setminus B_{ktp}} z_{iktp} \geq \left\lceil \frac{|B_{ktp}|}{|P|} \right\rceil . \tag{2.21}$$

   Observe that the number of customers that can be moved among the routes decreases with the number of depots. Inequalities (2.21) 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 [3], [1], [6]). Subtours elimination constraints are removed from the formulation and dynamically added. Priority and logical valid inequalities are added. 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 the symmetry breaking constraints and the fractional capacity cuts are not added. Finally, the branching rule is left to the MIP solver, instead of giving priority to the z variables.

2.5 Matheuristic for the $MDIRP$

We propose a matheuristic algorithm for the solution of the $MDIRP$ able to solve realistic–size instances. It is based on a 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 [50] 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

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

Set $\kappa = 0$;

Let $\overline{s}_{\mathcal{RMDIRP}}$ be a solution for the $RMDIRP$

Set $Ws_{\mathcal{RMDIRP}} = Bs_{\mathcal{RMDIRP}} = \overline{s}_{\mathcal{RMDIRP}}$; where $Ws_{\mathcal{RMDIRP}}$ is the worst $RMDIRP$ solution, whose cost is c_W and $Bs_{\mathcal{RMDIRP}}$ is the best $RMDIRP$ solution whose cost is 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 (2.21)

end if

end for

Solve the $RMDIRP$ with diversification constraints , and let $\overline{s}_{\mathcal{RMDIRP}}$ be the corresponding solution.

if $c_{s_{\mathcal{RMDIRP}}} < c_{Bs_{\mathcal{RMDIRP}}}$ **then**

Set $Bs_{\mathcal{RMDIRP}} = \overline{s}_{\mathcal{RMDIRP}}$

end if

if $c_{s_{\mathcal{RMDIRP}}} > c_{Ws_{\mathcal{RMDIRP}}}$ **then**

$Ws_{\mathcal{RMDIRP}} = \overline{s}_{\mathcal{RMDIRP}}$

end if

$GAP = \frac{(c_B - c_W)}{c_B} 100$

$\kappa = \kappa + 1$

end while

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. Two types of routes are generated: *intra-cluster routes* and *inter-cluster routes*. In the first type, each route can visit only customers in the cluster, while in the second type each route can also visit borderline customers, i.e., customers not in the cluster, but close enough to it. 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 $\mathcal{R}$ to perform in each period and the quantity to deliver to each customer in each of the selected routes. This quantity can be different than the one used to generate the routes.

We now describe these three phases in detail.

2.5.1 Clustering phase

The aim of this phase is to partition the set of customers I into a set of clusters $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{|P|}\}$. 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 [22], where a qualitative definition of critical customers is proposed. We propose a quantitative measure, computed on the basis of 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:

$$CL_i = \alpha R_i + (1 - \alpha) M_i, \tag{2.22}$$

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 $DC_i^p = c_{TST}^p - c_{TST\setminus\{i\}}^p$ 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 $I\setminus\{i\}$ from depot $p \in P$ ($c_{TST\setminus\{i\}}^p$). The values of R_i and M_i in (2.22) 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 and b_{pi} a binary variable equal to 1 if customer i belongs to the cluster C_p, and 0 otherwise. Then, the model is formulated as follows:

$$\min \quad \sum_{(p,i)\in E} c_{pi}\, b_{pi} \tag{2.23}$$

$$\text{s.t.} \quad \sum_{i\in I} b_{pi} \le CC \qquad \forall\, p \in P \tag{2.24}$$

$$\sum_{p\in P} b_{pi} = 1 \qquad \forall\, i \in I \tag{2.25}$$

$$\sum_{i\in I} CL_i b_{pi}/CC \le TC \qquad \forall\, p \in P \tag{2.26}$$

$$b_{pi} \in \{0,1\} \qquad \forall\, p \in P,\ \forall i \in I. \tag{2.27}$$

The objective function (2.23) minimizes the total distance between customers and depots. Constraints (2.24) enforce the total number of customers in each cluster to be within the maximum number CC. Constraints (2.25) impose that each customer is assigned to exactly one cluster. Constraints (2.26) enforce the average critical level of each cluster to be not greater than the maximum value TC. Constraints (2.27) define variables b_{pi}.

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

2.5.2 Routing construction phase

The aim of the second phase is to generate a restricted number of routes for the $MDIRP$ on the basis of the clusters generated by model (2.23)–(2.27) and on the basis of some replenishment policies. Two classes of routes are generated: *intra-cluster* and *inter-cluster routes*. The first class is composed of all routes serving customers that are in the same cluster, while the second class is composed of all the routes that serve also 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 to the corresponding policy in [3], 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 the cluster C_p and for each time period, 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 the cluster C_p and for each time period, 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, 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 having not 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, the replenishment quantity $\hat{y}_{it}(g_4)$ is computed as follows:

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

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 Cap: C, $C/2$ and $C/3$. This meets the aim of generating different routes. In the following, we will 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: $TSPR^g_{C_p} \leftarrow INTRARG\,(H, g, C_p, Cap)$.

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 2.1).

- 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 set on the basis of 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 2.1).

In the following, we will 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: $TSPR^g_{Bord_p \cup C_p} \leftarrow INTERRG\,(H, g, Bord_p \cup C_p, Cap)$.

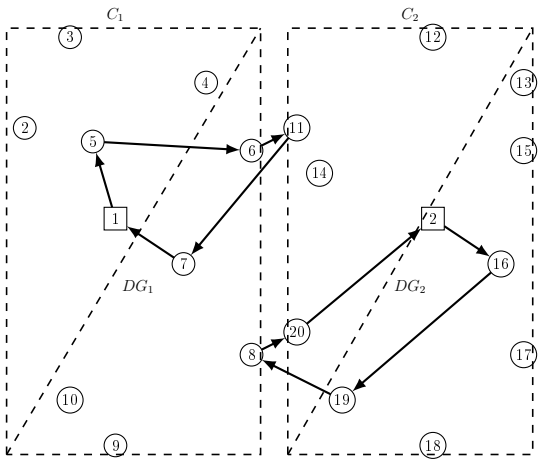

Figure 2.1: Examples of rectangular convex hull and inter-cluster routes.

A set R of delivery routes is built from the union of the set of all the intra–cluster routes and of the set of all the inter–cluster routes, excluding all

dominated routes.

2.5.3 Optimization phase

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

- c_r: cost of route $r \in R$
- 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:

$$\min \quad \sum_{t\in T}\sum_{r\in R} c_r\ n_{rt} \tag{2.28}$$

$$\text{s.t.} \quad Inv_{i,t} = Inv_{it-1} + \sum_{r\in R} m_{irt-1} - d_{it-1}, \quad \forall i \in I, \forall t \in T \cup \{H+1\} \tag{2.29}$$

$$\sum_{r\in R} m_{irt} + I_{it} \leq U_i, \quad \forall t \in T, \forall i \in I \tag{2.30}$$

$$\sum_{i\in I} m_{irt} \leq C\ n_{rt}, \quad \forall r \in R, \forall t \in T \tag{2.31}$$

$$m_{irt} \leq C\ a_{ir}, \quad \forall i \in I, \forall r \in R, \forall t \in T \tag{2.32}$$

$$\sum_{r\in R} n_{rt} \leq M, \quad \forall t \in T \tag{2.33}$$

$$Inv_{it} \geq 0, \quad \forall i \in I, \forall t \in T \cup \{H+1\} \tag{2.34}$$

$$m_{irt} \geq 0, \quad \forall i \in I, \forall r \in R, \forall t \in T \tag{2.35}$$

$$n_{rt} \in \{0,1\}, \quad \forall r \in R, \forall t \in T. \tag{2.36}$$

The objective function (2.28) minimizes the total routing cost. Constraints (2.29) define the inventory level at each customer i at each time period t. Constraints (2.30) enforce the maximum inventory level of each customer i at each time period t to be not greater than U_i. Constraints (2.31) enforce the total amount delivered with each route r in time period t to be within the vehicle capacity, while constraints (2.32) 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 (2.33) enforce the number of routes used in each time period t to be within the fleet size M. Finally, constraints (2.34)–(2.36) define the decision variables.

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

Algorithm 2 Pseudo-code for our matheuristic

$R := \emptyset$

Phase 1: clustering

for $i \in I$ **do**

compute parameter R_i, M_i, $CL_i = \alpha R_i + (1-\alpha) M_i$

end for

solve the **clustering** problem (2.23)–(2.27), $\mathcal{C} \leftarrow Clustering\,(I_{MDIRP})$

Phase 2: Routing construction

for each cluster $C_p \in \mathcal{C}$ **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 R: $R := R \cup \{r_i^p\}$

end for

for each replenishment policy g_1, g_2, g_3, g_4 **do**

for each capacity $Cap \in \left\{Q, \frac{Q}{2}, \frac{Q}{3}\right\}$ **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 $TSPR^g_{C_p} \leftarrow INTRARG\,(H, g, C_p, Cap)$

update R: $R = R \cup TSPR^g_{C_p}$

determine $TSPR^g_{Bord_p \cup C_p} \leftarrow INTERRG\,(H, g, C_p, Cap)$

update R: $R = R \cup TSPR^g_{Bord_p \cup C_p}$

end for

end for

end for

Phase 3: Optimization

eliminate from R all the dominated routes

solve the **route-based** $MDIRP$ formulation (2.28)–(2.36)

2.6 Computational results

The branch-and-cut algorithm described in Section 2.4 and the matheuristic described in Section 2.5 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*.

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 [3]; 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. 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 2.4.

2.6.1 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 cost 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 2.1. 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 [29]. Finally columns **Optimal MH** and **Optimal EX** describe the number of instances closed to optimality into each group for the two approaches. Note that each group is made up of five instances.

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 2.1: Performance on the Single-Depot, Single-Vehicle IRP

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 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.

2.6.2 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 [3], 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 (2.28)–(2.36), 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 2.2 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 (2.28)–(2.36).

Instance	N.Clust.	$\|C_i\|$	N. Routes	Time (s)	Cost	MHIRP GAP %	Instance	N. Clust.	$\|C_i\|$	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 2.2: Matheuristic performance for instances with $H = 3, 6$

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 $H = 6$.

2.6.2.1 Branch-and-cut performance

Table 2.3 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 **Sub-tours**, **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 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 2.3: Branch–and–cut performance for instances with $H = 3, 6$

2.6.2.2 Performance comparison

In this section the comparison between the results of the matheuristic algorithm and the branch–and–cut is presented. Table 2.4 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 2.4 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 $H = 6$

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 2.4: Performance comparison

without increasing the number of route–variables in formulation (2.28)–(2.36). 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.

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 2.4, 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.

2.6.2.3 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 table is 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 $\frac{Sol.inter-c}{Solution\ Routes}$.

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 Tables 3.2, 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.

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

2.7 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. Future research could be devoted to improve the branch-and-cut algorithm and to study the $MDIRP$ in order to better adapt this problem to the real cases.

2.8 Acknowledgments

The work by Leandro C. Coelho was partly supported by the Canadian Natural Sciences and Engineering Research Council under grant 2014–05764. This support is gratefully acknowledged.

Chapter 3

A Variable MIP Neighborhood Search for the Multi-attribute Inventory Routing Problem

Abstract: In this Chapter, a mathematical formulation for the Multi–attribute Inventory Routing Problem ($MAIRP$) is proposed. More precisely, we extend the Multi–depot Inventory Routing Problem ($MDIRP$) in order to consider the multi-product case with a heterogeneous fleet of vehicles and explicit constraints for the route duration. The $MAIRP$ is a NP–hard problem more complex than the classical Inventory Routing Problem (IRP). Nevertheless, it captures a lot of the features that can be found in real applications of the VMI setting. We introduce a hybrid algorithm to solve it, in which several Mixed Integer Programming (MIP) models are solved to explore the neighborhoods of a Variable Neighborhood Search (VNS) scheme applied to the $MAIRP$. The impact is to accelerate the resolution process with respect to other approaches used for the MIP standalone. We design several neighborhoods that are based on the features of the problem, and we present computational results on classical benchmark instances.

Keywords: Variable Neighborhood Search, Mixed-Integer Programming, In-

ventory Routing Problem.

3.1 Introduction

In this Chapter a rich variant of the IRP is introduced, the so–called Multi-attribute Inventory Routing Problem ($MAIRP$). The classical Inventory Routing Problem (IRP) combines inventory management policies and vehicle routing operational plans within a unique and coordinate scheme. In the case of the $MAIRP$, a vendor has to deliver a group of products from different origins (a set of depots) to a set of customers with a heterogeneous fleet of vehicles, while limiting the time duration of routes. The objective of the problem is to jointly minimize depots and customers inventory costs and the transportation cost, while satisfying the customers' demands and avoiding the stock–out. Since the $MAIRP$ is a general case of the IRP, it is a NP–hard problem. A hybrid algorithm based on the Variable Neighborhood Search (VNS) method is designed, where the neighborhoods' exploration is performed through a branch–and–cut algorithm working on a small portion of the whole search space. This methodology was already sucessfully applied to the optimization problem related to the inventory and distribution of cash for automated teller machines by Larrain et al. [43]. The authors called this approach *Variable MIP Neighborhood Search* ($VMNS$). The name is self-explanatory, so we decided to maintain the same definition. An extensive overview of the IRP and its industrial applications could be found in [2], while a deep literature review is described in [25]. Different algorithms were proposed to solve the IRP: exact methods are presented in [3], [23] and [24], heuristic algorithms are described in [14], [29] and [47], while hybrid algorithms can be found in [43] and [16].

The proposed hybrid algorithm starts with an initial solution that is built by solving the $MAIRP$ mathematical formulation by the means of a branch–and–cut algorithm that is stopped when an integer feasible solution is found

within a time limit. At this point the local search phase starts by solving a sequence of neighborhoods in order to improve the current solution obtained. The algorithm continues to alternate the two phases as explained in the following sections. The exploration of the neighborhoods is performed through $MIPs$ according to a given order. The idea to solve the local search using a MIP is not new, it was explored in the literature by [5] and [23], while the orderly exploration is inspired by the Variable Neighborhood Descent search [44]. Some similarities can be found with other MIP heuristics like fix-and-relax [35], restrict-and-relax [42] and relaxation induced neighborhood search [31]. The last technique is similar to our methods and it is used in commercial solvers.

The remainder of the chapter is organized as follows: in Section 3.2 the problem is described while in Section 3.3 the mathematical formulation of the problem is provided. In Sections 3.4 and 3.5 the hybrid $VMNS$ is detailed and in Section 3.6 some preliminary computational results are shown, followed by conclusion in Section 3.7.

3.2 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 suppose that G is an Euclidean graph, in which the triangular inequality holds. We partition the set V in such a way that $V = D \cup I$ where $D = \{1, 2, \ldots, m\}$ is the set of depots and $I = \{m + 1, m + 2, \ldots, n\}$ is the set of customers. A set of different products $P = \{1, 2, \ldots, p\}$ has to be delivered from the depots to the customers, over a finite and discrete time horizon H. Let $T = \{1, \ldots, H\}$ be corresponding set of time periods. A set $K = \{1, 2, \ldots, k\}$ of vehicles with different capacities Q_k is available to perform the deliveries, and let α_k be the variable cost for distance unit. Each vehicle can be assigned only to one depot $d \in D$ in each

time period $t \in H$. Each customer $i \in I$ can be served by different vehicles in the same period $t \in H$, travelling from different depot $d \in D$. A maximum inventory level U_i and a given starting inventory level I_{ip0} for each product $p \in P$ are associated with the customers. We assume that they are both integer. A given starting inventory level I_{id0} is associated with each depot for each product $p \in P$. The inventory level capacity at the depots is unlimited. In each time period t, a deterministic integer demand d_{ipt} for each customer i and each product p must be satisfied. An amount r_{dpt} of each product p in each depot d and for each period t is available, it is also integer. The inventory level of each customer i cannot be negative in each time period $t \in T \cup \{H+1\}$. A non-negative cost c_{ij} and a travelling time τ_{ij} are associated with each edge $(i,j) \in E$. A non-negative unit inventory holding cost h_{ip} is associated to customer i and product p and a non-negative unit inventory holding cost h_{id} for depot d and product p is considered. The service time is the same for each customer and it is denoted by ω, while the maximum duration time of each route is M.

The aim is to determine:

1. the set of customers to be served by a depot in each time period;
2. the vehicles to assign to a supplier by considering that each vehicle cannot be shared among suppliers in the same period;
3. the quantity of each product to deliver to each customer at each time period;
4. the set of routes to execute that minimize the total inventory and transportation costs.

The replenishment plan is subject to the following constraints:

- inventory level at each customer can never exceed its maximum capacity;

- no stock-out can occur at the customers;
- each vehicle can perform only one route in the time period, that starts and ends at the same depot;
- vehicle capacity cannot be exceeded;
- maximum time allowed for each route cannot be exceeded;
- connections between depots are forbidden (a vehicle cannot transit from a depot to another depot in a route);
- the split is allowed only in the sense that a customer could be serviced in the same time period by different vehicles that start their tour from different depots;
- for each period only one vehicle can start its tour from one depot;
- maximum inventory levels of the customers and vehicle capacities are shared among different products.

3.3 Mathematical formulation

We present a mathematical formulation of the problem. We introduce the following notation. Let $\delta(S)$ be the set of edges (i, i') incident to the vertices $i \in S \subset V$ (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 sub–set of customers. Our mathematical formulation is based on the following variables:

- I_{ipt}: inventory level at customer $i \in I$ at the end of period $t \in T$ for the product $p \in P$;
- I_{dpt}: inventory level at depot $d \in D$ at the end of period $t \in T$ for the product $p \in P$;

- y_{ipktd}: quantity to deliver to customer $i \in I$ of product $p \in P$ in period $t \in T$ by vehicle $k \in K$ starting the tour from depot $d \in D$;
- x_{ijktd}: binary variable equal to 1 if vehicle $k \in K$ travels directly from vertex $i \in V$ to vertex $j \in V$ in period $t \in T$ starting the tour from depot $d \in D$ and 0 otherwise;
- z_{iktd}: binary variable equal to 1 if vehicle $k \in K$ visits customer $i \in I$ in period $t \in T$ starting the tour from depot $d \in D$ and 0 otherwise;
- z_{dktd}: binary variable equal to 1 if vehicle $k \in K$, located in depot $d \in D$, starts the tour from depot $d \in D$ in period $t \in T$ and 0 otherwise;

The mathematical formulation is described by (3.1)–(3.19).

$$\min \quad \sum_{i\in I}\sum_{p\in P}\sum_{t\in T} h_{ip}I_{ipt} + \sum_{d\in D}\sum_{p\in P}\sum_{t\in T} h_{dp}I_{dpt} + \sum_{t\in T}\sum_{(i,j)\in E}\sum_{k\in K}\sum_{d\in D} \alpha_k c_{ij} x_{ijktp} \tag{3.1}$$

$$\text{s.t.} \quad I_{dp,t+1} = I_{dpt} - \sum_{k\in K}\sum_{i\in I} y_{ipktd} + r_{dpt} \quad \forall t \in T, \forall p \in P, \forall d \in D \tag{3.2}$$

$$I_{ip,t+1} = I_{ipt} + \sum_{k\in K}\sum_{d\in D} y_{ipktd} - d_{ipt} \quad \forall t \in T, \forall p \in P, \forall i \in I \tag{3.3}$$

$$\sum_{p\in P}\sum_{k\in K}\sum_{d\in D} y_{ipktd} + \sum_{p\in P} I_{ipt} \leq U_i \quad \forall t \in T, \forall i \in I \tag{3.4}$$

$$\sum_{i\in I}\sum_{p\in P} y_{ipktd} \leq Q_k \; z_{dktd} \quad \forall t \in T, \forall d \in D, \forall k \in K \tag{3.5}$$

$$y_{ipktd} \leq Q_k \; z_{iktd} \quad \forall i \in I, \forall t \in T, \forall p \in P, \forall k \in K, \forall d \in D \tag{3.6}$$

$$\sum_{i\in I}\sum_{p\in P} y_{ipktd} \geq z_{dktd} \quad \forall t \in T, \forall d \in D, \forall k \in K \tag{3.7}$$

$$z_{bktd} = 0 \quad \forall t \in T, \forall d \in D, \forall k \in K, \forall b \in D\,(p \neq b) \tag{3.8}$$

$$\sum_{d\in D} z_{dktd} \leq 1 \quad \forall t \in T, \forall k \in K \tag{3.9}$$

$$\sum_{d\in D} z_{iktd} \leq 1 \quad \forall t \in T, \forall k \in K, \forall i \in I \tag{3.10}$$

$$\sum_{k\in K} z_{dktd} \leq 1 \quad \forall t \in T, \forall d \in D \tag{3.11}$$

$$\sum_{(i,j)\in E} \tau_{ij} x_{ijktd} + \sum_{i\in I} \omega z_{iktd} \leq M \quad \forall k \in K, \forall t \in T, \forall d \in D \tag{3.12}$$

$$\sum_{j\in I, j<i} x_{ijktd} + \sum_{j\in I, j>i} x_{ijktd} = 2z_{iktd} \qquad \forall i \in V, \forall t \in T, \forall d \in D, \forall k \in K \tag{3.13}$$

$$\sum_{(i,j)\in E(S)} x_{ijktd} \leq \sum_{i\in S} z_{iktd} - z_{uktd} \qquad \forall S \subseteq V, |S| \geq 2, \forall t \in T, \forall k \in K, \forall d \in D, u \in S \tag{3.14}$$

$$x_{ijktd}, x_{djktd}, x_{jdktd} \in \{0,1\} \qquad \forall i,j \in I, \forall t \in T, \forall d \in D, \forall k \in K \tag{3.15}$$

$$I_{ipt} \geq 0 \qquad \forall i \in I, \forall t \in T, \forall p \in P \tag{3.16}$$

$$I_{dpt} \geq 0 \qquad \forall d \in D, \forall t \in T, \forall p \in P \tag{3.17}$$

$$y_{ipktd} \geq 0 \qquad \forall i \in I, \forall t \in T, \forall p \in P, \forall k \in K, \forall d \in D \tag{3.18}$$

$$z_{iktd}, z_{dktd} \in \{0,1\} \qquad \forall i \in I, \forall t \in T, \forall d \in D, \forall k \in K. \tag{3.19}$$

The objective function (3.1) minimizes the total inventory and routing costs at the customers and suppliers. Constraints (3.2)-(3.3) define the inventory at the suppliers and customers. Constraints (3.4) impose the maximum inventory level at the customer. Constraints (3.5)-(3.6) are capacity constrains on the vehicles. Constraints (3.7) establish that vehicle $k \in K$ starts the tour from depot $d \in D$ in period $t \in T$ if an amount of products is delivered to some customers by this vehicle. Constraints (3.8) forbid connections between depots and constraints (3.9) impose that each vehicle can be located in only one depot. Constraints (3.10) enforce each customer to be visited with a route starting from one depot at most in a given period $t \in T$ and with vehicle $k \in K$. Constrains (3.11) allow only one vehicle can leave a depot in a time period. Constraints (3.12) impose that duration of the route cannot be exceeded. Constraints (3.13) and (3.14) control the degree of the vertices and prohibit subtours, respectively. Finally, constraints (3.15)–(3.19) define the integrality and non-negative conditions for the variables.

3.4 Variable MIP Neighborhood Search

In this section we describe the two main tools used for solving the problem. In Section 3.4.1 the branch-and-cut phase is described, while in Sections 3.4.2 and 3.5 the local search scheme is detailed.

The $VMNS$ is a matheuristic algorithm that uses a MIP to explore neighborhoods. In our algorithm, the $MIPs$ are based on a mathematical formulation with routes variables that reproduces the problem of the (3.1)-(3.19) formulation, in which some variables are fixed and new constraints are added.

The algorithm is based on two different phases called *routines* in the following, that are executed alternately. These *routines* differ mainly as to their structure: the first one implements a branch–and–cut algorithm that is the main framework of the Variable Neighborhood Search scheme and it is named *BCR*, while the second one implements the matheuristic algorithm described in the second Chapter as the main tool used in the Variable Neighborhood Search scheme, and it is referred to as *LSR*. In the following the two routines are described in details, while the structure of the $VMNS$ is presented in the next sections.

3.4.1 Branch–and–cut Routine

The branch–and–cut solves the problem obtained by removing constraints (3.14) from problem formulation (3.1)- (3.19), and by adding only the violated subtour elimination constraints for each period $t \in T$, for each vehicle $k \in K$ and for each depot $d \in d$. They are added dynamically using the exact separation procedure described in [46]. They were introduced considering a given $u \in S$, for which the following condition is valid: $u = argmax_i\{\bar{z}_{iktd}\}$, where $\bar{z}_{iktd}$ is the value of variable z_{iktd} in the current LP relaxation. At each tree node, the violated (3.14) are checked and added to the current sub-problem that is then optimized. If no violations are identified, branching occurs at the

current node. No priority variables are defined for the branching strategy. Moreover, in order to improve the quality of the root node lower bound of the branch–and–cut tree, the following valid inequalities are added to the initial LP of formulation (3.1)- (3.19):

1. *Priority inequalities:*

$$z_{iktd} \leq z_{dktd}, \quad \forall i \in I, \forall k \in K, \forall d \in D, \forall t \in T \tag{3.20}$$

   These valid inequalities were presented by [3] to improve the performance of the proposed branch-and-cut for the single–vehicle IRP.

2. *Logical inequalities:*

$$x_{idktd} + x_{diktd} \leq 2\ z_{iktd}, \quad \forall i \in I, \forall t \in T, \forall d \in D, \forall k \in K \tag{3.21}$$

$$x_{ijktd} \leq z_{iktd}, \quad \forall i, j \in I, \forall t \in T, \forall d \in D, \forall k \in K \tag{3.22}$$

   Inequalities (3.21) impose that if depot d is the predecessor or the successor of customer i in the route executed in period t by vehicle k starting from depot d, then customer i has to be visited in period t by vehicle k starting from depot d. Inequalities (3.22) impose that if customer j is the successor of customer i in the route performed in period t by vehicle k starting from depot d, then customer i has to be visited in period t by vehicle k starting from depot d. They are inspired by the logical cuts discussed in [37] and [38].

3.4.1.1 Initial solution

The performance of the branch–and–cut algorithm described in Section 3.4.1 is sped up by finding an initial solution computed as follows. We solve a relaxed formulation of (3.1)-(3.19), in which the routing part is dropped from the objective function, while the total routing cost is estimated by adding the

following term $\sum_{t\in T}\sum_{k\in K}\sum_{d\in D}\sum_{i\in I} c_{di}z_{iktd}$, where c_{di} is the cost of the edge $(d,i) \in E$. Moreover, all the routing constraints (3.12)-(3.14) and the corresponding variables x_{ijktd} are dropped from the mathematical formulation. In the relaxed version of the problem, the constraint (3.12) is substituted by a new constraint that contains an estimation of the route duration:

$$\sum_{i\in V}(MD_d + \omega)z_{iktd} \leq M \qquad \forall t \in T, \forall k \in K, \forall d \in D \tag{3.23}$$

where MD_d is the medium travelling time. A set of feasible clusters is defined on the basis of the values of variables z_{iktd} for each $k \in K$, for each $t \in T$ and for each $d \in D$. For each of such clusters, the optimal TSP tour is found and the corresponding cost is added to the optimal objective value of the relaxed formulation.

3.4.2 Local Search Routine

This routine builds several neighborhoods whose exploration helps in finding feasible solutions that are better than the initial one provided by the BCR. In order to explore quickly the neighborhoods, the mathematical structure of the MIPs that are solved repeatedly must be easier than the one presented in the previous Section. In light of the preliminary computational study and of the analysis presented in Chapter 2, all the following observations were considered in designing the mathematical structure of the MIPs solved in the $VMNS$ scheme:

- due to the decomposable structure of the $MDIRPs$, clustering techniques are helpful to reduce the complexity of such problems;
- a good way to explore neighborhoods in a Local Search framework is to use a matheuristic algorithm providing new solutions with a small computational effort in the neighborhood of a given solution;

- Route–based formulations are more effective to approach multi–attribute $MDIRPs$ than arc–based formulations.

On the basis of the above, we decided to embed the matheuristic algorithm described in Chapter 2 into the VNS scheme. Obviously, the integration implies some modifications and adaptations that will be described in the following, where the main steps of the matheuristic are re-called:

1. *clustering phase* : an integer linear programming model is solved to obtain a partition of the set of customers into a set of clusters, one for each depot;

2. *Routing construction phase*: a set $\mathcal{R}$ of routes is built for the clusters generated in the first phase. The routes are generated by varying the vehicle capacity and by considering different replenishment policies. Two types of routes are generated: *intra-cluster routes* and *inter-cluster routes*.

3. *Optimization phase*: a binary linear programming model with route variables is optimally solved to obtain a feasible solution of the $MAIRP$.

The matheuristic was not implemented straight as it to explore each neighbourhood. Indeed, clustering and routing construction phases are computed only once at the beginning of the whole $VMNS$ execution, in order to generate fixed clusters and to obtain a good set of feasible routes. The route–based mathematical formulation is solved each time a neighbourhood is explored by adding tailored constraints that accelerate finding an improved solution. In the following short description of the necessary adjustments of the matheuristic algorithm is provided.

3.4.2.1 Routing construction phase

The aim of the second phase is to generate a restricted number of routes for the $MDIRP$ on the basis of the clusters generated by model (2.23)–(2.27) and on the basis of different replenishment policies. As explained in section 2.5.2,

two classes of routes are generated: *intra-cluster* and *inter-cluster routes*. The unique difference with respect to the $MDIRP$ relies in the presence of multiple products at the same customer. However, since the maximum inventory level at each customer and the vehicle capacity are resources that are shared among different products, no change occurs in the route generation procedure described in Chapter 2. In fact, the demand of each customer, the initial inventory levels and the quantity scheduled by each policy are not considered product-by-product, but they are aggregate during the route generation process. Despite the fact that model (3.1)-(3.19) accounts for these quantity separately, the route generation procedure is able to determine a set of routes that are still effective for the multi–product case. In the replenishment policies described in section 2.5.2 it is necessary to make the following substitutions: $d_{it} = \sum_{p \in P} d_{ipt}$; $\hat{I}_{it} = \sum_{p \in P} I_{ipt}$ and $\hat{y}_{it} = \sum_{p \in P} y_{ipt}$. The procedure for building the routes remains unchanged, excepted for capacity parameter Cap. In the new problem a heterogeneous fleet of vehicles is considered, so the intra–cluster routes are generated on the basis of the different capacities by setting with $Cap \in \{C_k, C_k/2, C_k/3; \quad \forall k \in K\}$, while the inter–cluster routes are determined by considering only maximum vehicle capacity, namely with $Cap = max\{C_k; \quad \forall k \in K\}$.

3.4.2.2 Optimization phase

In the third phase, route–based mathematical formulation is solved with the set of routes $\mathcal{R}$, and a feasible $MAIRP$ solution is found in each neighborhood. A partition of $\mathcal{R}$ is defined according to the vehicle capacities, as $\mathcal{R} = \{R_1, R_2, ..., R_k\} \quad \forall k \in K$, where each subset R_k is composed by the routes in which vehicles with capacity C_k are used. We introduce the following parameters that are indexed in the set of routes, $r \in \mathcal{R}$:

- c_r: cost of route $r \in R$,
- t_r: travelling time of route $r \in R$,

- a_{ir}: binary parameter equal to 1 if customer i is served in route r, 0 otherwise

and the following decision variables:

- m_{irtpd}: quantity of product p shipped to customer i in route r in time period t starting from depot d;
- Inv_{itp}: inventory level of product p at customer i at the end of time period $t \in T \cup \{H+1\}$;
- Inv_{dtp}: inventory level of prodcut p at depot d at the end of time period $t \in T \cup \{H+1\}$;
- n_{rtd}: binary variable equal to 1 if route r is used in time period t starting from depot d, 0 otherwise.

The problem is then formulated as follows:

$$\min \quad \sum_{t\in T}\sum_{r\in R}\sum_{d\in D} c_r\ n_{rtd} + \sum_{t\in T}\sum_{p\in P}\sum_{d\in D} h_d\ Inv_{dtp} + \sum_{t\in T}\sum_{p\in P}\sum_{i\in I} h_i\ Inv_{itp} \tag{3.24}$$

$$\text{s.t.} \quad Inv_{d,t+1,p} = Inv_{dtp} - \sum_{r\in R}\sum_{i\in I} m_{irtdp} + r_{dtp}, \quad \forall d \in D, \forall t \in T, \forall p \in P \tag{3.25}$$

$$Inv_{i,t+1,p} = Inv_{itp} + \sum_{r\in R}\sum_{d\in D} m_{irtdp} - d_{itp}, \quad \forall i \in I, \forall t \in T, \forall p \in P \tag{3.26}$$

$$\sum_{r\in R}\sum_{p\in P}\sum_{d\in D} m_{irtpd} + \sum_{p\in P} I_{itp} \leq U_i, \quad \forall t \in T, \forall i \in I \tag{3.27}$$

$$\sum_{i\in I}\sum_{p\in P} m_{irtpd} \leq C_k\ n_{rtd}, \quad \forall r \in R_k, \forall R_k \in R, \quad \forall t \in T, \forall d \in D \tag{3.28}$$

$$m_{irtpd} \leq C_k\ a_{ir}, \quad \forall i \in I, \forall r \in R_k, \forall R_k \in R, \quad \forall t \in T, \forall p \in P, \forall d \in D \tag{3.29}$$

$$\sum_{r \in R} n_{rtd} \leq 1, \qquad \forall t \in T, \forall d \in D \tag{3.30}$$

$$\sum_{r \in R_k} \sum_{d \in D} n_{rtd} \leq 1, \qquad \forall t \in T, \forall R_k \in R \tag{3.31}$$

$$\omega |r| x_{rtd} + t_r x_{rtd} \leq M, \qquad \forall t \in T, \forall r \in R, \forall d \in D \tag{3.32}$$

$$Inv_{itp} \geq 0, \qquad \forall i \in I, \forall t \in T, \forall p \in P \tag{3.33}$$

$$Inv_{dtp} \geq 0, \qquad \forall d \in D, \forall t \in T, \forall p \in P \tag{3.34}$$

$$m_{irtpd} \geq 0, \qquad \forall i \in I, \forall r \in R, \forall t \in T, \forall d \in D, \forall p \in P \tag{3.35}$$

$$n_{rtd} \in \{0, 1\}, \qquad \forall r \in R, \forall t \in T, \forall d \in D. \tag{3.36}$$

The objective function (3.24) minimizes the total inventory and routing cost. Constraints (3.25) and (3.26) define the inventory level at each customer and depot. Constraints (3.27) enforce the maximum inventory level of each customer i at each time period t to be not greater than U_i. Constraints (3.28) enforce the total amount delivered with each route r in time period t to be within the vehicle capacity, while constraints (3.29) 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 (3.30) and (3.31) enforce the number of routes used in each time period t to be within the fleet size and manage the split. Constraints (3.32) impose that the travelling time and the service time do not exceed the total duration for each route. Finally, constraints (3.33)–(3.36) define the decision variables.

3.5 VMNS scheme

The main scheme of the procedure is described in the following. The following steps are executed only once during the whole $VMNS$:

- the relaxed version of the initial problem is solved within a time limit α (as explained in 3.4.1.1), in order to provide an initial $MAIRP$ feasible solution;

- the clustering and route generation phases are computed in order to build all the necessary input sets for the algorithm (see section 3.4.2.1);

Secondly, the branch-and-cut routine (BCR) is started and solved until a new integer solution is found or a time limit β is reached. At this point, the local search routine (LSR) is executed: different neighborhoods of the current incumbent solution are explored in search of an improvement. The neighborhoods are ordered according to a given rule that is illustrated in the following Sections. A maximum time for this routine is established as well (γ). At the end, the algorithm comes back to the BCR that restarts from where it was left with a new initial solution, that is equal to the best improvement found in the LSR . For the total algorithm a time limit is imposed (δ).

In the following we reported the outline of the algorithm for the BCR and the LSR . We introduce the following notation: a solution for the MIP problem is represented as a vector Y with all the variable, the set $N = \{1, 2, \ldots, n\}$ indicates the set of the Neighbourhoods that have been explored. In general a neighbourhood is obtained applying an operator to the current improvement Y. (Y, n) indicates that operator n is applied to solution Y. In our case, the operator consists of setting a subset of variable to the value they have in the current improvement and leaving the other free. In this way it is possible to solve a restricted MIP that corresponds to explore a neighbourhood in the LSR. We use the MIP formulation (3.24)-(3.36) to explore the LSR; while formulation (3.1)-(3.19) is used in the BCR routine. The main scheme of the algorithm is represented in Figure 3.1.

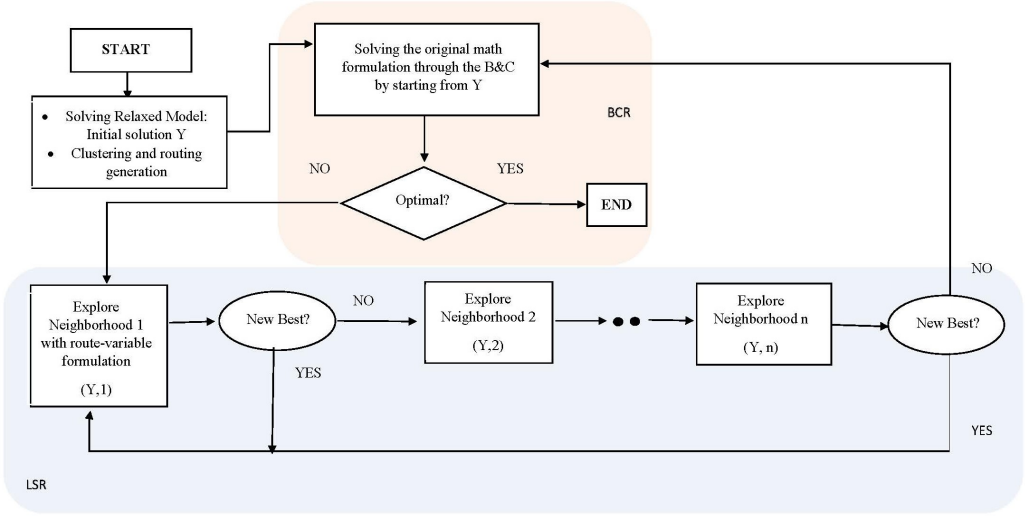

Figure 3.1: $VMNS$ scheme

The pseudo-code of the algorithm is reported in Algorithm 3.

Algorithm 3 Pseudo-code $VMNS$

solve the RELAXED MAIRP until time limit α is reached $\rightarrow Y^0$
solving clustering and routing generation procedure for building the set $\mathcal{R}$
solve the BCR with initial MIP-Start Y^0 until time limit β is reached
while stopping time limit δ is not met **do**
  solving LSR (Y, γ)
  solving BCR (Y, β)
end while

BCR : in this phase the $MAIRP$ is solved. At the beginning, a feasible solution is found through a relaxation of the main MIP and it is passed as initial solution to the branch-and-cut algorithm. After the LSR is executed, the new initial solution for the branch-and-cut is represented by the best im-

provement found in the local search phase.

LSR : in this phase different neighborhoods are explored around a given solution Y (the best solution found in the BCR, when $n = 1$). If a new improvement is found in the current n, the procedure is re-started from the beginning and the new solution is explored applying all the neighborhoods from $n = 1$. If no improvements are found, the incumbent is explored with the following neighborhoods. The procedure ends if the time limit is reached or if all the neighborhoods are explored without finding a new improvement. In order to save the time, instead of solving each neighborhood to optimality, the solver stops as soon as it finds an improvement of the current solution Y or when the time limit is reached. The LSR is described in detail by Algorithm 4.

Algorithm 4 Pseudo-code LSR

Input: starting solution Y , $n = 1$
while $n = |N|$ or time limit γ is reached **do**
  explore (Y, n)
  if a new improvement is found **then**
    n=1
  end if
  if no improvement is found **then**
    n=n+1
  end if
end while

3.5.1 Neighborhood description

We developed four different neighborhoods for the $VMNS$. They are divided in two main categories: simple neighborhoods, that consider only one

feature of the problem at a time, and complex neighborhoods, that combine different features. The list is reported below:

1. **Depot**: select a depot $\overline{d}$ and fix all the variable values related to it, while all the remaining variables rest free for each $d \in D \setminus \{\overline{d}\}$. The exploration is repeated for each depot within the same neighbourhood.

2. **Time**: select a period $\overline{t}$ and fix all the variable values related to it, while all the remaining variables rest free for each $t \in H \setminus \{\overline{t}\}$. The exploration is repeated for each period within the same neighbourhood.

3. **Route**: select a route $\overline{r}$ and fix all the variable values related to it, while all the remaining variables rest free for each $r \in R \setminus \{\overline{r}\}$. The exploration is repeated for each route contained in the last improved solution in the same neighbourhood.

4. **Depot-Depot**: select two depots $\overline{d_1}$ and $\overline{d_2}$ and fix all the variable values related to them, while the other variables remain free. The exploration is repeated for each combination of these depot indices in the neighbourhood;

5. **Route-Route**: select two routes $\overline{r_1}$ and $\overline{r_2}$ and fix all the variable values related to them, while the other variables are free. The exploration is repeated for each combination of these indices contained in the last improved solution in the neighbourhood;

All the neighborhoods where a large number of variables is set are expected to be explored faster than the other ones, because they have a small size. If the size of neighborhoos is very small it is possible to have no improvement of the objective function, while if it is large a long time is required to find a good feasible solution that improves the incumbent. The aim is to find a good trade-off.

3.6 Computational results

In this section computational results on the $MAIRP$ instances are presented and discussed . The algorithm was 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 IBM CPLEX 12.6.1 Concert Technology with default parameters to solve the $MIPs$. Concorde solver (*http://www.math.uwaterloo.ca/tsp/concorde.html*) is used to find the optimal TSP solution. The performance of the algorithms is evaluated on a set of instances derived from the benchmark set of [23], adapted for the $MAIRP$. We maintain the original depot as the first depot and we generate the remaining $d-1$ depots randomly. Our data set is composed of instances with a number of customers ranging from 5 to 25, while the number of depots ranges from 2 to 4 based on the size of the instance. The time horizon H is equal to 3. In order to generate heterogeneous vehicle instances, we consider a fleet of 3 vehicles whose capacity is obtained by reducing/increasing the original capacity by $\frac{1}{3}$ up to 1. Instances are labelled as *SnNdDhHpP*, where S indicates the instance number, N is the number of customers in the instance, D is the number of depots, H is the number of periods and P is the number of products. The parameters of the heurisitc are set as follows: $\alpha = 240s$, $\beta = 100s$, $\gamma = 1200s$ and $\delta = 7200$. Table 3.1 reports provided by the $VMNS$ algorithm. Column **Instance** describes the instance name, column **Solution** reports the best solution found by the algorithm, column **Improvements** introduces the total number of improvements found by the procedure, column **Time(s)** introduces the computational time in seconds and column **GAP%** describes the Cplex-GAP related to the final improvement in the BCR . The $VMNS$ works well on small instances, while the risk to be trapped in local optimal solutions increases with the size of the instances. Given the difficulty of the problem, the results demonstrate that the heuristic algorithm is effective. The algorithm is able to

approach small and medium instances maintaining the medium GAP around the 16%, while only 6 instances are closed to optimality. We can observe that the algorithm does not improve the quality of the initial solution when it is executed on small instances, even using different neighborhood search strategies. This case occurs whenever the initial solution is optimal. In the following Table, we assess the behaviour of the local search in terms of improvements provided in each neighborhood's exploration.

Instance	Solution	Improvements	Time(s)	GAP%
1n5d2h3p3	217.11	0	13	0.00
1n10d2h3p3	5613.74	10	7200	21.16
1n15d2h3p3	5917.00	7	7200	23.31
1n20d3h3p3	8354.77	4	7200	27.44
1n25d4h3p3	10659.5	3	7200	22.55
2n5d2h3p3	218.15	0	6	0.00
2n10d2h3p3	6152.04	10	7200	25.67
2n15d2h3p3	5812.28	8	7200	20.00
2n20d3h3p3	9801.62	4	1492	0.00
2n25d4h3p3	11596.20	2	7200	31.40
3n5d2h3p3	221.98	0	7	0.00
3n10d2h3p3	5487.40	10	7200	19.86
3n15d2h3p3	6117.99	7	7200	22.59
3n20d3h3p3	11625.2	3	7200	16.96
3n25d4h3p3	10581.10	2	7200	24.56
4n5d2h3p3	216.89	0	5	0.00
4n10d2h3p3	7201.62	8	7200	18.36
4n15d2h3p3	6583.35	5	7200	25.59
4n20d3h3p3	10261.00	2	7200	7.98
4n25d4h3p3	9923.42	2	7200	27.95
5n5d2h3p3	228.05	1	571	0.00
5n10d2h3p3	7080.10	7	7200	13.23
5n15d2h3p3	6035.47	5	7200	19.27
5n20d3h3p3	9708.79	6	7200	7.44
5n25d4h3p3	10659.50	3	7200	31.59
Average		5	5555.76	16.27

Table 3.1: VMNS computational results

In Table 3.2, the number of improvements found in each neighborhood is shown. The Table is organized as follows: each column reports the number

of improvements obtained by exploring the following neighborhoods, named as **Depot**, **Time**, **Route**, **Depot-Depot**, **Route-Route**, respectively. The best improvements are achieved whenever a neighborhood involves a modification in the depot set or in the route set. This is due to the fact that the neighborhoods Depot and Route operate directly on the more sensitive features of the problem. Neighborhood *Route* is more effective than the one named *Depot*, since it modifies the assignment of a subset of customers to a given depot. Indeed,

Instance	Depot	Time	Route	Depot-Depot	Route-Route
1n5d2h3p3	0	0	0	0	0
1n10d2h3p3	2	0	8	0	0
1n15d2h3p3	2	2	2	1	0
1n20d3h3p3	0	0	0	0	4
1n25d4h3p3	1	0	1	0	1
2n5d2h3p3	0	0	0	0	0
2n10d2h3p3	1	0	6	0	3
2n15d2h3p3	3	1	4	0	0
2n20d3h3p3	0	0	0	0	4
2n25d4h3p3	1	0	1	0	0
3n5d2h3p3	0	0	0	0	0
3n10d2h3p3	2	2	6	0	0
3n15d2h3p3	3	0	4	0	0
3n20d3h3p3	0	1	0	0	2
3n25d4h3p3	1	0	1	0	0
4n5d2h3p3	0	0	0	0	0
4n10d2h3p3	1	0	6	0	1
4n15d2h3p3	2	0	3	0	0
4n20d3h3p3	1	0	1	0	0
4n25d4h3p3	1	0	1	0	0
5n5d2h3p3	0	0	1	0	0
5n10d2h3p3	2	2	3	0	0
5n15d2h3p3	1	0	4	0	0
5n20d3h3p3	1	2	0	0	3
5n25d4h3p3	1	0	0	1	1
Average	1.04	0.4	2.08	0.08	0.76
Total	26	10	52	2	19

Table 3.2: Neighbourhoods computational results

the effectiveness of the *Depot* is about 24%, while the one of *Route* is 48%. Due the small number of depots with respect to the number of customers that arises in the proposed instances, involving two routes in the neighborhood's

exploration provides a larger number of improvements than the ones offered by other combinations. In general, a single operator improves a lot the quality of the solution because it does not imply fixing a large number of variables in the current solution, so that the size of the neighborhood is large. The following figure summarizes the incidence of each neighborhood:

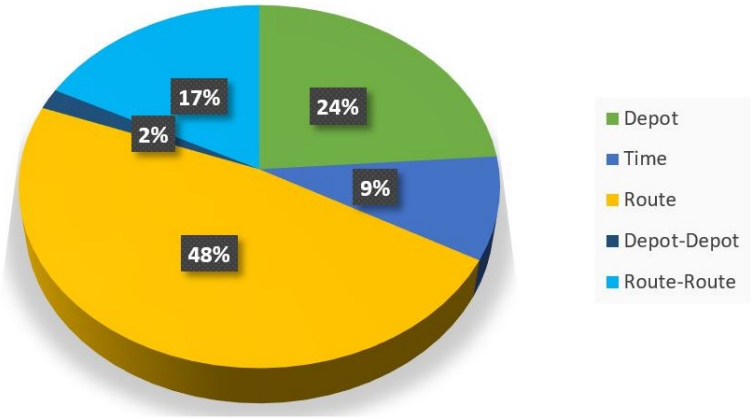

Figure 3.2: Neighborhood incidence

The Gaps provided by CPLEX remain still large due to the weakness of the lower bound provided by the LP relaxation. This occurs especially when the number of customers, products and depots is large. Therefore, to assess the performance of the $VMNS$ on classical IRP instances and to evaluate well the performance of the proposed algorithm, we executed additional test on the $MDIRP$ variant and on classical IRP, that refers to the case of the single-depot and single-vehicle problem.

The data set used in the additional test is composed of 25 instances with up to 25 customers, with an horizon $H = 3$ and inventory cost h_i in $[0.1, 0.5]$, (the data set is a sub set of the one used in *Chapter 2*). 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 2 hours was imposed to the $VMNS$ for solving the

problem. The table is organised as follows: column **Instance** describes the instance name, column **Solution** reports the best solution found by the algorithm, column **Improvements** introduces the total number of improvements found by the procedure, column **Time(s)** introduces the computational time in seconds, column **GAP%** describes the CplexGAP related to the final improvement in the *BCR*, column **B&C sol.** describes the solution value obtained by the branch–and–cut detailed in Section 2.4, while columns **B& C Time(s)** and **B&C GAP(%)** report its computational time and the performance quality, respectively.

Instance	Solution	Improv.	Time(s)	GAP%	B&C sol.	B&C Time(s)	B&C GAP%
1n5d2h3	1148.80	0	120.00	0.00	1148.80	89.71	0.00
1n10d2h3	3840.09	4	7200	25.71	2177.98	21700	10.91
1n15d2h3	4398.72	10	7200	26.68	4588.88	21700	27.35
1n20d3h3	3811.38	4	7200	35.56	3263.25	21700	19.32
1n25d4h3	4632.66	3	7200	46.54	4203.45	21700	31.93
2n5d2h3	956.24	0	68	0.00	956.24	434.76	0.00
2n10d2h3	2427.66	8	7200	6.82	2427.66	21700	2.11
2n15d2h3	2302.13	10	7010	0.00	2302.13	20955	0.00
2n20d3h3	4079.90	3	7200	33.18	4395.27	21700	20.66
2n25d4h3	7485.21	2	7200	44.22	4570.13	21700	35.61
3n5d2h3	1801.02	0	2.18	0.00	1801.02	360.51	0.00
3n10d2h3	1651.54	4	650	0.00	1651.54	1854.78	0.00
3n15d2h3	3075.69	5	5430	0.00	3075.69	6602.50	0.00
3n20d3h3	3456.70	3	7200	12.13	3399.42	21700	14.21
3n25d4h3	5364.83	2	7200	43.18	4127.82	21700	33.47
4n5d2h3	1425.39	0	3.47	0.00	1425.39	1537.42	0.00
4n10d2h3	2249.30	2	331	0.00	2249.30	1994.27	0.00
4n15d2h3	8955.30	4	7200	31.15	2481.13	21700	20.00
4n20d3h3	6163.29	4	7200	32.98	4501.08	21700	20.78
4n25d4h3	6253.37	2	7200	35.97	5722.99	21700	42.14
5n5d2h3	1808.40	0	240	0.00	1808.40	60.09	0.00
5n10d2h3	1982.90	2	454	0.00	1982.90	734	0.00
5n15d2h3	2454.81	7	7200	13.15	2912.07	21700	37.69
5n20d3h3	5226.28	4	7200	35.67	4338.40	21700	23.86
5n25d4h3	4921.22	2	7200	24.22	6214.81	21700	56.78
Average		3	4892.34	17.88		14404.92	15.87

Table 3.3: Performance on the Multi-depot IRP

The *VMNS* and the branch–and–cut are able to provide an optimal so-

lution for 10 instances. Note that the time limit for the two algorithms is different: $VMNS$ has a time limit of 2 hours while the alternative algorithm runs with 6 hours of time limit. The medium Cplex GAP for the $VMNS$ is around 17.88%, while for the branch–and–cut is around 15.87%. The $VMNS$ is able to find solution as good as the branch–and–cut in smaller computational time. In conclusion, we can state that the $VMNS$ is much effective in solving the $MDIRP$ than the branch–and–cut. On the other hand, it is not competitive with the matheuristic described in the previous chapter, because the instances have an homogeneous fleet of vehicles. This characteristic generates lots of equivalent $MDIRP$ solutions, that are not faced by the $VMNS$. The average number of improvements for each instance is equal to 3, where the effectiveness of the *Depot* neighborhood is about 32% and is about 31% for the *Route-Route* neighborhood. There are no improvements for the *Depot-Depot* neighborhood. Probably, more specific neighborhood are needed in order to produce most competitive results for the $MDIRP$ special case.

In order to investigate the behavior of the $VMNS$ on the single–attribute IRP we test the algorithm on a data set composed of 50 instances with up to 50 customers, with an horizon $H = 3$ and inventory cost h_i in $[0.1, 0.5]$, (the data set is derived by [3]). Instances are labelled as $absXnY$, where X is the instance number and Y is the customer number. The data concerning the supplier and the unit inventory costs are not considered. Numerical results are shown in Table 3.4. This table is organized as follows. Column **Instance** provides the instance name. Columns **VMNS** and **TimeVMNS (s)** provide the cost of the $VMNS$ solution and the corresponding computational time, respectively. Columns **EX** and **TimeEX (s)** report the cost of the best solution obtained within the time limit allowed with an exact approach (values with asterisk are optimal) and the corresponding computational time, respectively. Finally, column **Gap %** provides the gap between the two algorithms, computed as $GAP\ \% = \frac{VMNS-EX}{EX}100$. These results were provided by [29].

Instance	VMNS	TimeVMNS (s)	EX	TimeEX (s)	Gap %
abs1n5	1309.92	0.02	1309.92*	0.16	0.00
abs1n10	2040.29	0.03	2040.29*	0.23	0.00
abs1n15	2403.84	11.20	2403.84*	0.28	0.00
abs1n20	2687.02	7.60	2687.02*	0.31	0.00
abs1n25	3277.94	16.00	3277.94*	1.64	0.00
abs1n30	4261.6	34.99	4261.6*	3.78	0.00
abs1n35	4038.32	35.80	4038.32*	7.73	0.00
abs1n40	4740.68	35.70	4740.68*	129.00	0.00
abs1n45	5010.06	101	5010.06*	35.16	0.00
abs1n50	5173.26	2.00	5173.26*	110.66	0.00
abs2n5	1051.28	0.05	1051.28*	0.58	0.00
abs2n10	2442.47	4.00	2442.47*	0.81	0.00
abs2n15	2432.22	6.30	2432.22*	1.74	0.00
abs2n20	2934.18	12.00	2934.18*	3.61	0.00
abs2n25	3545.46	29.80	3545.46*	5.94	0.00
abs2n30	3990.9	45.30	3990.9*	17.23	0.00
abs2n35	4085.74	56.40	4085.74*	24.81	0.00
abs2n40	4572.27	40.00	4572.27*	100.94	0.00
abs2n45	4623.22	62.00	4623.22*	40.64	0.00
abs2n50	5414.25	158.70	5414.25*	244.03	0.00
abs3n5	1942.2	0.20	1942.2*	0.95	0.00
abs3n10	2136.22	4.00	2136.22*	2.22	0.00
abs3n15	2904.46	7.60	2904.46*	4.72	0.00
abs3n20	3152.3	8.91	3152.3*	8.42	0.00
abs3n25	3704.66	15.70	3704.66*	13.91	0.00
abs3n30	4450.7	29.64	4450.7*	22.19	0.00
abs3n35	4935.18	116.30	4935.18*	31.44	0.00
abs3n40	4946.02	93.00	4946.02*	45.63	0.00
abs3n45	5203.44	85.00	5203.44*	62.42	0.00
abs3n50	5526.59	139.20	5526.59*	170.8	0.00
abs4n5	1409.22	0.10	1409.22*	1.31	0.00
abs4n10	1908.46	0.50	1908.46*	3.31	0.00
abs4n15	2315.44	3.00	2315.44*	7.23	0.00
abs4n20	3334.21	6.00	3334.21*	15.63	0.00
abs4n25	3449.65	64.00	3449.65*	24.49	0.00
abs4n30	3729.76	67.00	3729.76*	50.69	0.00
abs4n35	3850.56	50.00	3850.56*	44.66	0.00
abs4n40	4141.62	89.00	4141.62*	62.48	0.00
abs4n45	4809.82	122.00	4809.82*	100.09	0.00
abs4n50	5644.14	202.60	5644.14*	157.09	0.00
abs5n5	1192.64	2.00	1192.64*	2.45	0.00
abs5n10	2295.09	5.00	2295.09*	4.52	0.00
abs5n15	2455.21	9.66	2455.21*	9.56	0.00
abs5n20	3440.98	21.20	3440.98*	18.13	0.00
abs5n25	3793.42	18.00	3793.42*	25.28	0.00
abs5n30	3715.86	25.34	3715.86*	37.56	0.00
abs5n35	4117.26	28.80	4117.26*	55.44	0.00
abs5n40	4581.28	100.65	4581.28*	97.00	0.00
abs5n45	4629	161.44	4629*	263.33	0.00
abs5n50	5357.4	64.50	5357.4*	361.45	0.00
Average		45.85		46.5	0.00

Table 3.4: Performance in the Single-Depot, Single-Vehicle IRP

The $VMNS$ provides an optimal solution for all the instances with a computational time that is as good as the one required by the exact solver. All the instances are solved to optimality in less than 3 minutes. In conclusion, we can state that the $VMNS$ is effective in solving also the classical single attribute IRP.

3.7 Conclusion

In this chapter we introduce a rich variant of the classical IRP: the Multi-attribute IRP. The problem is very challenging from the computational point of view, as well as it is very interesting for applications in real cases. We introduce a complex algorithm for solving it, that is based on two main phases: a branch-and-cut algorithm and a local search. The approach is inspired by the work of [43] that successfully applied it for solving real industrial instances for a particular variant of IRP. The $VMNS$ is expected to short the computational time with respect to other exact approaches. This is shown by applying the $VMNS$ both to the $MDIRP$ and to the classical IRP. The results demonstrate that the proposed method is competitive also in solving special cases of the $MAIRP$. The algorithm is flexible and it can be easily adapted for solving other classes of distribution problems. The fact that it relies on mathematical formulation helps the solving process also for big instances, nevertheless the procedure to generate new and different neighborhoods is also simple, because it consists in imposing new constraints. The algorithm described in the Chapter is quite flexible and competitive, its "double" nature allows to solve medium instances in a reasonable computational time providing good quality solution. The LSR accelerates the search mechanism of an exact solver, when it is careful embedded into the search–tree of a branch–and–bound method. The $VMNS$ represents a very inspiring search tool for future investigations and developments.

Chapter 4

Vendor-Management Inventory in the Nanostores: the last-mile delivering in Surabaya city

Abstract: Vendor Managed Inventory (VMI) is a coordination paradigm in the integrated management of the supply-chain that tries to optimize simultaneously inventory and routing. In this system, the decisions about timing and level of customer's replenishment are determined by the supplier that is supposed to have the complete knowledge about the customers' need, in order to avoid stock-outs. In this work, the approach presented in the Chapter 2 is applied to a logistic network in the supermarket distribution industry. The system comprises a distribution centre and several retails located in Surabaya, a port city on the Indonesian Island of Java. A detailed analysis of the context and some computational results are presented.

Keywords: Vendor Management Inventory, Clustering, Mega-city, Nanostores

4.1 Introduction

The aim of this chapter is to investigate and asses the convenience of the application of the VMI system in managing the supply chain of a specific *FMCG Company* segment of customers in Surabaya city, focusing on the last-mile deliveries in the urban environment. Generally, the last leg of the supply chain is often less efficient, and it usually represents up to 30% of the total cost for moving goods. The last mile problem also includes the challenge of making deliveries in urban areas where retail stores, restaurants, and other merchants in a central business district often contribute to congestion and safety problems. Moreover, in the last decades it was registered a significant population move from towns to big cities. It is expected that the tendency will grow in the next future. This situation produces the increase of traffic and congestion into dense areas of the cities, as a consequence. The efficiency of last-mile deliveries is a big challenge for lots of companies that every day needs to apply operational strategies in order to obtain a good trade-off between the customer satisfaction and the transportation costs.

The **FMCG Company** is an American multinational consumer goods corporation, operating in the hyper/super market chain, who serve customers of different sizes and typologies including Nanostores, that represent a big portion of the Company business. It produces and distributes a wide range of cleaning and personal care products and includes also food, snacks and beverage in its portfolio. The Company operates in four different big sectors (Beauty, Grooming and Health Care, Fabric & Home Care, and Baby, Feminine & Family Care) selling its products in approximately 180 countries. The distribution channels are different for size and typology, so they are represented by a large variety of retailers: mass merchandisers, grocery stores, drug stores, hyper and super markets, distributors, baby stores, beauty stores, e-commerce, high-frequency stores, pharmacies and nanostores. The Company distinguishes its territories of action in emerging markets (Asia, Africa and Latin-America) and devel-

oped markets (North America and Western Europe). Generally, the *FMCG Company* supply chain can be considered a multiple–echelon system: finished products are carried from the production plant to the Plant Distribution Centre (PDC), from there to the Regional Distribution Centre (RDC) and afterwards also to the Retail Distribution Centre. This part of the supply chain is managed completely by the Company (it is underlined in figure 4.1).

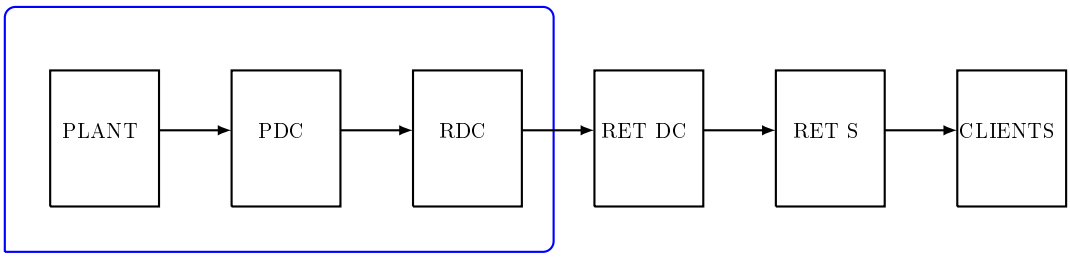

Figure 4.1: *FMCG Company* supply chain: a multi–echelon system

The freight that moves from Retail Distribution Centre ($RETDC$) and Retail Shop (RET S) is not under the Company control. The $RETDC$ level usually assumes various configurations, it could coincide with different types of distributors like wholesalers, large retailers, discounts or distribution centres of third logistics parties. Actually the outbound logistics is an outsourcing activity: the Company does not own a fleet of vehicles for deliveries. It usually signs long-terms contracts with different transport companies. The agreements are not the same in all the geographical contexts (e.g. the presence or the absence of particular infrastructures, particular law of the region), because it is necessary to take into account specific needs and environment requirements, all these aspects influence the prices and the transport modes. As consequence, there is not a homogeneous way for moving freight in the last part of the supply–chain.

As all the consumer packaged goods companies, *FMCG Company* faces

hard challenges for distributing its products within cities, especially for big and mega cities. This term is referred to a city with a total population in excess more than 10 million inhabitants [21]. Only the Asian continent hosted half of the world's 29 mega-cities at the end of 2015. This number is expected to grow in the future. The mega-city' s big challenge is frequently considered as traffic congestion, but in reality it is not the only infrastructural problem. There are lots of issues related to power and water supply, energy and resources consumption, crime, health and safety, air and noise pollution.

In this work, the main focus is about the mega city network structure for considering opportunity and challenges in the consumer good shop replenishing activity. The high population density favours the birth of millions of small stores located in the urban areas. These traditional channel stores are independent, family-owned stores (like kiosks, neighbourhood stores, grocery and convenience stores) and they are typical of the emerging markets. *FMCG Company* usually refers to these stores as **Nanostores**, that are characterized by high levels of fragmentation and small order quantities (they are represented by the *RET* S level on figure 4.1).

Nanostore sector is considered today the largest potential channel for growth together with E-commerce for the Company. In a world where more and more people are moving into large city environments, and with a growing number of logistical challenges arising in such contexts, one of the Company's aim is to bring its products to the *Nanostore* owners/consumers at an affordable last mile cost (which accounts for 30-70% of the total supply chain cost). For an in depth overview about the *Nanostore* classification the reader can refer to the *Appendix A*.

This analysis aims to look at **the potential value of re-applying the** VMI **concept in the Nanostore channel within a big–city context and understand the value such a set-up can generate vis-a-vis the current set-up**. In fact, a VMI approach will help in optimizing the order sizes, routing

efficiencies and on-shelf-availability, likely this will have a positive effect on sales as well. As explained in *Section 2.1*, the VMI represents a win-win strategy for the integrated management and optimization of two different stage activities in the supply chain (storage and delivery), that was largely investigated in the past and successfully applied by lots of companies. For an in depth overview of the field, the reader can refer to [18] and [25] for mathematical methods and algorithms, and to [2] for the industrial applications. The chapter is organized as follow: in the Section 4.2 the logical flow followed to develop the analysis is described, in Section 4.3 the Surabaya city context and the features of the network are deeply investigated, in Section 4.4 the scope of the application is described and in Section 4.5 all the computational results and their impact on the business are shown.

4.2 Methodology

In order to provide satisfying results for the Company, the methodology employed to complete the project and to analyse the case–study in Surabaya city is divided in three different consequent steps and it is reported in figure 4.2.

In the **STEP 1**, the data set provided by the Company is analysed. All the data available are pre-processed and purified for eliminating errors or inconsistency in the information. After this activity, the context of analysis is deeply investigated: the data are localized in their geographical context and different studies are conducted with the aim to select the best method to build the graph network (the main starting point of the optimization process). The details about the activities of the STEP 1 are described in Section 4.3.

In the **STEP 2**, we define a general heuristic framework for solving the problem, that is summarized below. The main idea is to divide the city into little districts to manage the distribution problem. Each district presents a

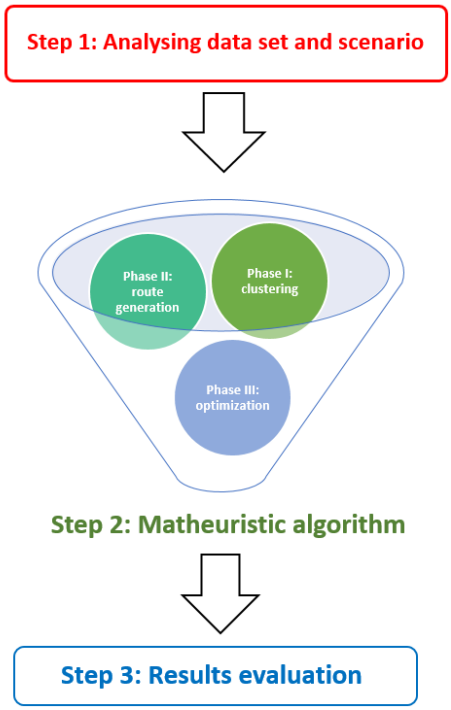

Figure 4.2: Logical flow in the analysis

hub-shop, in a central position, that stores and deliver freight to the final customers. The hub-shop have to cover the demand of the entire district. In order to face the problem in the best way it was decided to divide the heuristic methodology in three distinct phases:

1. in the first phase, we define a good procedure to create the districts. In particular, a procedure for aggregating the customers into clusters is considered, in which a single depot is responsible for the second level distribution. Clustering is performed on the base of different parameters (distance depot-customer, demand level, inventory level);

2. in the second phase, a set of feasible delivery routes is generated;

3. in the third phase an IRP mathematical formulation is solved through an algorithm that create a good and feasible schedule in order to decide:

when to deliver freight to each customer, how much to deliver and which is the best routing planning for delivery, with the final aim of minimizing the total transportation costs.

The details of the algorithm are previously described in *Chapter 2*.

In the **STEP 3**, the results collected applying the matheuristic algorithm are discussed in order to evidence the possible improvements for the business in delivering products in the last-mile, applying a VMI strategy at the place of the traditional Customer Managed Inventory.

4.3 A general overview

As said before, this section presents a general overview about the Surabaya city structure and all the features related to the *FMGC* case study, in order to focus the attention on the most important elements. The context of analysis and the network are deeply described in the following.

4.3.1 The context of analysis

Surabaya is a port city on the Indonesian island of Java. The total metropolitan area presents a population of 6.484.206 (in 2010), making Surabaya now the third largest metropolitan area in Indonesia, after Greater Jakarta and Greater Bandung. Also if it is not formally defined as Mega-city, it is characterized by the same challenges and affected by same traffic and congestion problems.

FMCG Company operates in this emerging market delivering its products to all the retailers located in the urban space, that belong to the category of *Nanostores*. The distribution network consists in one central depot (or vendor) and a set of retailers (or customers) localised in the Surabaya urban area. The demand of the customers occurs continuously at a rate. Standard deliveries are carried out by a heterogeneous fleet of trucks. The data set is composed of a central depot and a list of **947** shops to be replenished, characterized

Figure 4.3: Surabaya City

by different features: geographical coordinates, addresses, delivery volumes, actual scheduled days of replenishment, orders quantities, for a total horizon of one month. Some data are also available for the three typologies of vehicles (different for their capacity) used for deliveries and for their logistic costs. The depot is in a central position, the shop locations are represented below:

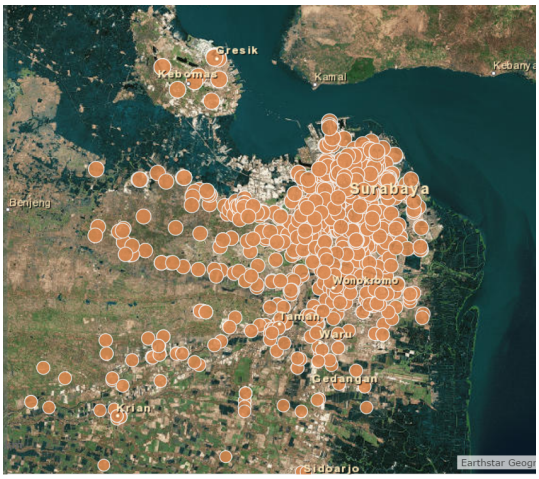

Figure 4.4: Geographical Nanostores' positions

As can be seen, Nanostores are located very near each other. In the current situation the city is divided into 12 ***Zones*** by the distributors, that perform the last-mile deliveries in outsourcing for the Company. Each zone is made up of base units, called ***Areas***. The deliveries and the truck assignment are

organized on area base. The total number of areas is 49. Areas and Zones are defined at the beginning of the activities and they are considered fixed. The difference between Zone and Area is presented in the following figure:

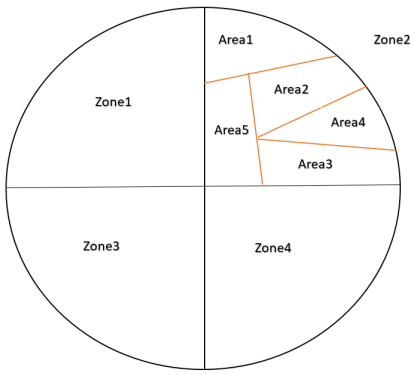

Figure 4.5: Difference between "Area" and "Zone"

The 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 into his territory and to organize the vehicle for the delivery. The information flow between the vendor and the customers is only using limited ICT systems and optimizations. Indeed, the salesman usually uses traditional ways for collection orders like by phone or directly speaking with the Nanostore owners. Areas are different for extension and size, the number of customers in each area is heterogeneous. The details are reported in the table below:

Zone	N. Customers	N. Areas	Cardinalities
Zone1	32	4	3,1,26,2
Zone2	28	9	1,1,18,3,1,1,1,1,1
Zone3	49	6	5,6,1,2,28,7
Zone4	59	10	2,2,21,2,1,1,5,1,1,23
Zone5	57	7	1,7,1,42,1,1,4
Zone6	135	16	46,1,2,6,3,2,21,10,1,8,5,2,4,2,2,20
Zone7	135	21	9,1,4,22,40,10,1,9,10,1,8,1,6,1,2,2,1,1,1,2,3
Zone8	138	21	6,33,4,5,14,7,9,6,2,23,3,6,7,3,2,2,1,1,2,1,1
Zone9	80	1	80
Zone10	51	1	51
Zone11	131	1	131
Zone12	52	9	3,4,1,1,19,22,1,1

Table 4.1: Actual number of customers into each Area

The aim of the analysis is to maintain the zones in delivery scheme and to re-organize the areas into each zone.

4.3.2 The network

In order to apply the matheurisitc algorithm presented in *Chapter 2* for testing the case-study, it is necessary to build a graph for representing the network of *Nanostores* and the street connections between them. A graph is usually composed by different elements: **vertices or nodes** (fixed points to refer), **arcs** (oriented connections between two vertices) and **edges** (not oriented connections between two vertices). When a graph is used to model an urban area, the arcs and edges correspond to the road segments, and vertices correspond to road intersections or customers to be served. The urban space is really complex to be modelled through a graph because it is necessary to consider different directions between two points, outward voyage, return trip and lots of intersections.

The data set contains the GPS coordinates of each Nanostore, so the position of each shop on the real map becomes a graph vertex. It is necessary to find a method to fix the edge/arcs, with the associated distance between

nodes and to understand which is the best way to measure the distances between two nodes. For the Surabaya city, the total number of *Nanostores* is 947, concentrated into a not so large urban space.

Considering an enlargement of the representation it is possible to notice that lots of *Nanostores* are often concentrated in the same area of the city, and in many cases also in the same streets.

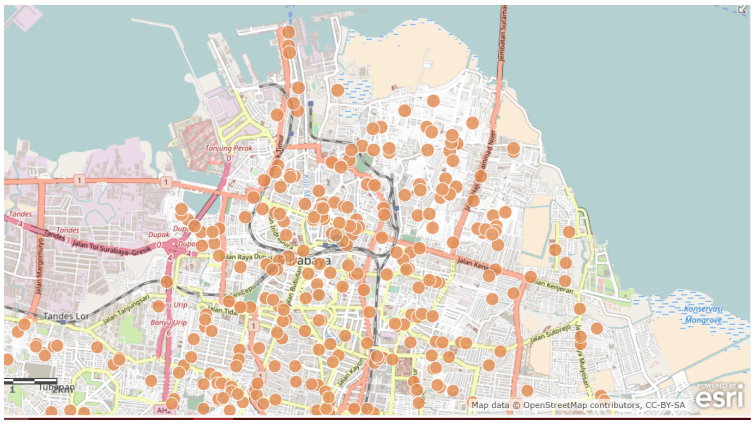

Figure 4.6: Nanostore in urban area: enlargement representation

In order to build a useful and plausible graph–network we need to consider different aspects:

- it is necessary to simplify the high complexity of the urban space as much as possible;
- it is necessary to guarantee that the triangular inequalities hold on our graph;
- the customers are positioned very close each other.

Starting from these evidences, the possibility to represent the network of Surabaya city through an *undirected graph* was investigated.

In general, when a graph is built on a real city, the normal procedure is to consider the shortest real route (shortest path) between the origin and the

destination of each arc/edge, taking into account one-way streets, deviations and all the components that are typical of the street urban network. In order to simplify the graph, we analyse the possibility of considering the **linear geodetic distance** between two points on the earth, at the place of the shortest-path, see [34].

The main difference between two methods is done by measurement precision and time consuming. In fact, while the first method gives in output the real distances between the points, the second sacrifices the precision for the fastness in the measurement. The necessary time amount to apply the second method is considerably shorter than the other. Both the methods guarantee the validity of the triangular inequalities.

In order to apply the best strategy, the analysis is focused to understand which is the real difference of measure in the case of Surabaya city. The main idea is to compute the distances through the two methods for a limited number of *Nanostores*, and to compare them in order to understand the committed error using the geodetic distance at the place of the shortest path. For this purpose, an on line software for mapping points and computing distances on the street network is used (*ArcGIS*). A set of 300 different distances between two nodes is chosen (completely random). For these points the shortest-path is computed through the on-line tool [1]. For the geodetic linear distance, considering two point A and B and their latitude (LAT) and longitude (LON), the following formula is used (see [34]):

$$\begin{aligned} Distance_{AB} = R * \arccos(\sin LAT(A) * \sin LAT(B) + \\ \cos LAT(A) * \cos LAT(B) * \cos LON(A) - LON(B)); \end{aligned} \tag{4.1}$$

where the Earth ray is:

$$R = 6731 \quad km \tag{4.2}$$

[1] *ArcGis* calculates it as the optimized distance routed by vehicles considering the specific rules for the cars

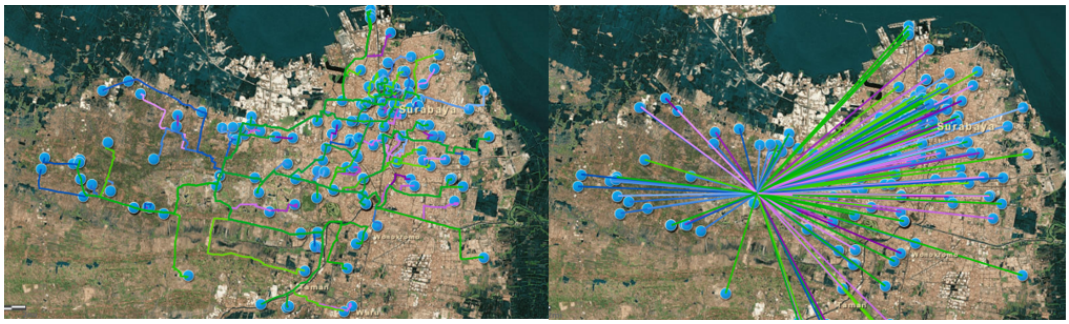

Figure 4.7: Difference between geodetic distance and shortest path in Surabaya area

For comparing the real distance (compute by the program) and the geodetic distance (compute with the formula), the **Mean Squared Percentage Error** ($MSPE$) is evaluated, that is the average error committed considering an estimator at the place of the real data observed. In this case the estimator is the geodetic distance. The $MSPE$ could be considered the measure of the quality of an estimator. It is obvious that values of the error closed to zero are good, because that means our semplification is very closed to the reality. If a set of n estimator $\hat{X}$ is considered; and X is the vector of observed values, the $MSPE$ of the predictor can be estimated as reported below (see [39]):

$$MSEP = \frac{\sum_{I} (\frac{\hat{X}_i - X_i}{X_i})^2}{|I|} \tag{4.3}$$

The $MSPE$ committed considering the Geodetic Distance (the estimator $\hat{X}_i$) at the place of the driving distance on the graph (observed value X_i) is equal to **0.49%**. Starting from the point that the committed error is minimum, we decide to measure the length of each arch/edge using the geodetic distance.

4.4 Managerial challenge and assumptions

As explained before, a large part of the work described in this thesis was carried out in the context of an industrial research project, aimed to reorganize the last-mile deliveries at *FMCG Company*, with a special emphasis on the customer deliveries to *Nanostores*. The reorganization of the distribution activities based on VMI model is the core of the project. The main output of the study is represented by the prototype of an algorithm that is able to schedule the deliveries in a certain horizon and to calculate the best routes to serve customers located into large urban spaces. It was realized starting from the scientific literature on $IRPs$ and best practices. In order to keep computational times within acceptable limits, a deterministic approach was implemented and deeply investigated in *Chapter 2*. After the theoretical phase, a practical analysis is developed on the Surabaya context, in order to investigate the real effectiveness of the algorithm. Through the research work, the Company wants to pursue the following objectives:

1. testing and evaluating the proposed model and algorithm with real data of the Company with a really large network of different customers in a Mega-City environment;

2. understanding the real value for the business in applying the VMI setting at the place of the classical Customer Managed Inventory in a mega-city context; applying also VMI in a retail channel where it has not been used so far;

3. comparing the use of a simple VMI framework with one where some cross–docking/little–depots are positioned into the city, in order to understand if the multi–depot approach could be useful for the business.

The data set provided by the Company is incomplete or it does not fully match with all the input parameters required by the algorithm. For this reason,

a series of initial *assumptions* are fixed before the analysis, after a preliminary discussion with the company's management people:

- the capacity of the central depot is supposed to be infinitive;
- the fleet is supposed homogeneous with the capacity of each vehicle assumed equal to 300 boxes (the medium size capacity of the vehicles used in reality by the Company);
- the driving time for each vehicle is assumed 8 hours/day;
- the maximum inventory level is defined equal to $\frac{3}{2}$ of the the medium daily demand for the customer into the horizon; actually the Nanostores often risk the stock-out, so their inventory capacity has to be not so large with respect to the demand;
- the initial inventory for each customer is defined equal to $\frac{1}{2}$ of medium daily demand for the customer into the horizon, defined like a small portion of the maximum inventory level;
- a series of mini–depots are randomly generated for each area in order to apply the clustering phase of the algorithm; also in this case their capacity is supposed infinitive.

All the other input data are extracted on the data set provided by the Company. Furthermore, the necessary adjustments were done in order to adapt the algorithm also at the single-depot case.

4.5 Computational results

In this Section, we illustrate the experiments aimed at evaluating the effectiveness of the approach just described and 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*. The performance of the algorithm is evaluated on a set of 12 instances (one for each Zone) derived from the data set for Surabaya city, with 28 to 138 customers. The number of depots increases from 3 to 11 according to the size of the instance. The time horizon H is 6. 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 to CPLEX for solving the mathematical model given by (2.28)–(3.36).

4.5.1 Clustering results

As explained in the previous sections, in the first step of the algorithm we re–define the customer's clusters for each Zone, trying to create clusters that are balanced from the cardinality and critical level point of view. For an in–depth overview of this aspect, the reader has to refer to *Section 2.5.1*. Table 4.5.1 provided the results and it is organized as follows. Column **Zone** provides the zone name, column **N.Customers** reports the number of customers into each Zone, column **N.Depots** shows the number of depots generated for each Zone, column **N.Areas** provides the number of new Areas generated in each Zone after the clustering phase and the column **Cardinalities** describes the number of customers contained into each Area. The table shows the number of the different areas we obtained in each zone. If we compare the results with the initial situation shown in the Table 4.1, it is evident that the new clusters are balanced and homogeneous. In the initial situation in which the Company operates, there are lots of inefficiencies, probably due to the mechanism through the deliveries are organized that makes the distribution and the territories so fragmented. In the case of the VMI setting, this aspect is managed by one entity that is able to optimize the delivery scheme, as a whole. We are able to

Zone	N. Customers	N. Depots	N. Areas	Cardinalities
Zone1	32	3	3	14,12,6
Zone2	28	2	2	14,14
Zone3	49	6	5	10,12,10,12,5
Zone4	59	3	3	23,13,23
Zone5	57	3	3	19,19,19
Zone6	135	10	10	14,14,14,14,14,14,14,14,9,14
Zone7	135	11	11	13,15,14,1,15,4,15,15,15,13,15
Zone8	138	11	11	12,15,15,15,15,10,5,15,6,15,15
Zone9	80	6	6	6,15,15,15,15,14
Zone10	51	3	3	19,13,19
Zone11	131	9	9	15,15,15,15,15,15,11,15,15
Zone12	52	3	3	18,16,18

Table 4.2: Clustering Results

eliminate also all the inefficiencies related to the presence of clusters made up of only one or two customers, for building a good set of feasible routes.

4.5.2 Route generation and final optimization results

In the second and in the third phase of the algorithm, a set of feasible routes is built, and it is used to solve an *IRP* model with route variables, in order to obtain the final scheduling for the deliveries and the minimization of the total kilometres covered by the vehicles. All the details about these two steps are described in *Sections 2.5.2 and 2.5.3*. Table 4.5.2 collects the results and it is organized as follows. Column **Zone** introduces the name of each zone, column **N. Routes** indicates the total number of routes generated by the algorithm; column **MHSol.(km)** describes the solution obtained from the algorithm, in terms of total kilometres routed for delivering freight in each Zone; column **Time(s)** reports the computational time for each instance and the column **GAP %** indicates the CplexGAP of the model (2.28)–(2.36).

In general, it is possible to observe that all the instances are closed to optimality, in a not so long computational time. The columns 3 and 4 underline

Zone	N. Routes	MHSol.(km)	Time(s)	TimeIRP(s)	Gap%
Zone1	778	346,99	64,82	3,58	0.00
Zone2	968	126,13	27,61	8,05	0.00
Zone3	1011	90,49	208,3	7,67	0.00
Zone4	4531	267,16	683,98	60,72	0.00
Zone5	3544	308,6	890	35,44	0.00
Zone6	4447	490	5759,21	39,65	0.00
Zone7	4735	534,86	4603,71	37,21	0.00
Zone8	4574	534,36	3719,53	35,46	0.00
Zone9	2887	196	3057,89	18,56	0.00
Zone10	2747	145,09	121,5	7,73	0.00
Zone11	4983	148,74	6000	43,7	0.00
Zone12	2721	136,49	187,74	20,17	0.00
AVERAGE	3161		2110,36	26,49	0.00

Table 4.3: Matheuristic Results

that the bigger part of the computational time is consumed in the routes generation phase, because the final optimization needs only few seconds in all the instances. If we compare the case-study results with the benchmark instances run in *Chapter 2* with $H = 6$, we can underline that the medium 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 medium number of routes generated is more than 5 times respect to the benchmark case. This extreme increase in the number of routes is a bit expected and obviously related to the particular shape of the network. Actually the customers are very near each other and they present small frequent orders, for the majority. The route generation phase is affected for the both factors, because it is driven by the clusters and the borderline customers (a big number in our case) and by the capacity of the vehicle. So the different combination of customers for possible feasible routes becomes copious. In spite of this aspect, the total computational time does not increase so much. The computational results demonstrated that the matheuristic maintain its effectiveness also in complex and real networks, because it is able to provide fast and good solutions also in real situations, that are more

complex that the literature benchmarks tested in *Chapter 2*.

4.5.3 Performances Comparison

In the following, a comparison between the actual situation in Surabaya city and the results obtained with our approach is provided. In particular, the total amount of covered kilometres by the vehicles in each Zone is compare with the amount computed by the algorithm that simulates the use of a VMI system. Table 4.5.3 shows the results and it is organized as follows. Column **Zone** introduces the name of each zone, **MHSol.(km)** describes the solution obtained from the algorithm, in terms of total kilometres routed for delivering freight in each Zone, column **MHSol.+MSEP** introduces the previous value with the correction described in *Section 4.3.2*; column **Actual(Km)** provides the total kilometres routed in Surabaya city with the Customer Managed Inventory approach (the classical one) and it is derived from the data set; finally the column **GAP%** describes the gap between the solution obtained with the VMI approach and the real data, computed as explained in the following:

$$GAP = (\frac{(MHSol+MSEP)-Actual(Km)}{Actual(Km)}) * 100$$

The last column shows that the matheuristic approach is able to find solutions better than 21,65% for the deliveries in term of total kilometres reduction. In general, the algorithm is able to find a better solution than the real situation for around the 92% of the instances. This is a valuable result, because it underlines the potential benefit for the business in applying a VMI system also in the last part of the supply chain, where deliveries are fragmented and not so easy to optimize. *It is possible to affirm that implementing a VMI approach could be an interesting investment to optimize the deliveries also in emerging markets where the big-cities are featured by the large presence of Nanostores.*

Zone	MHSol.(km)	MHSol.+MSEP	Actual (Km)	GAP%
Zone1	366,77	368,38	193,73	90,25
Zone2	126,13	126,41	140,19	-9,83
Zone3	90,49	90,74	203,75	-55,46
Zone4	267,16	267,99	226,04	18,55
Zone5	308,6	309,73	365,42	-15,24
Zone6	490	491,66	710,32	-30,78
Zone7	534,86	536,52	710,28	-24,46
Zone8	534,36	536,065	757,38	-29,22
Zone9	196	196,69	344,50	-42,90
Zone10	145,09	145,65	232,17	-37,26
Zone11	148,74	149,09	540,46	-72,41
Zone12	136,49	136,94	279,90	-51,07
AVERAGE				-21,65

Table 4.4: Comparing reality and simulated VMI

4.5.4 Single-Depot VMI vs. Multi-Depot VMI

In the previous analysis, the case of a VMI with a big single depot was considered, in order to reflect the actual situation in the city of Surabaya, where a big DC is situated in the urban area for the storage of the freight. According with the Company, the second aspect to investigate is what can happen if a VMI system with a series of mini-depots and/or cross-docking points is considered. The main question is understanding the convenience of this possibility (that faces lots of investments in infrastructures) comparing with the single-depot case. In this paragraph, we compare the results concerning the single-depot and the multi-depot case. For the second approach, we can use the algorithm of *Chapter 2* without necessity of any adjustment or adaptation. Table 4.5.4 shows the results and it is organized as follows. Column **Zone** introduces the name of each zone, column **SD-MH(Km)** reports the solution deeply described in the previous paragraph, column **MD-MH(Km)** describes

the total kilometres routed for the deliveries in the second scenario, column **GAP%** reports the gap between the two solutions that is computed as follow:

$GAP = (\frac{SDMH - MDMH}{SDMH}) * 100$

Zone	SD-MH(Km)	MD-MH(Km)	GAP%
Zone1	368,38	346,48	-5,94
Zone2	126,41	57,20	-54,74
Zone3	90,74	49,51	-45,43
Zone4	267,99	169,34	-36,80
Zone5	309,73	232,13	-25,05
Zone6	491,66	339,64	-30,91
Zone7	536,52	345,81	-35,54
Zone8	536,065	349,70	-34,76
Zone9	196,69	142,69	-27,45
Zone10	145,65	115,56	-20,65
Zone11	149,09	71,34	-52,14
Zone12	136,94	92,45	-33,49
AVERAGE			-33,05

Table 4.5: Single-depot VMI vs. Multi-depot VMI

Results underlines that using a configuration with multi points of replenishment can allow a further saving of around 33% in terms of kilometres covered by the vehicles. This aspect could be a good advice for the Company for deeper investigating this solution scheme during the analysis of a possible investment for introducing a VMI system into these emerging markets.

4.5.5 Loading Performances

Another aspect to consider in analysing the simulation of VMI system is related to the loading performances. As said before, in this moment the Company usually uses contracts with third parties for the freight delivering.

The data set reports an heterogeneous fleet composed of three different types of vehicles: the smallest with a capacity of 100 boxes, the medium with a capacity of 300 boxes (which one we use in the analysis) and the bigger with a capacity of 500 boxes. We consider obviously the medium size. Table 4.5.5 shows the results and it is organized as follows. Column **Zone** describes the name of the zone, column **Del.Quantity/week** reports the number of boxes delivered in each zone during the horizon (in this case the horizon is one week), column **N. Trip** introduces the number of necessary routes the vehicle has to perform in each zone for satisfy the demand, column $\frac{DQ}{Trip}$ shows the medium number of boxes for each trip, column **LF300** describes the load factor for the vehicles with capacity of 300 boxes in each trip while column **LF100** introduces the load factor for the vehicles with capacity of 100 boxes in each trip. The load factor is computed as follows:
$LF = \frac{\frac{DQ}{tripS}}{VehicleCapacity}$.

Zone	**Del.Quantity/week**	**N.Trip**	$\frac{DQ}{Trip}$	**LF300**	**LF100**
Zone1	289	5	57,8	0,19	0,58
Zone2	516	9	57,33	0,19	0,57
Zone3	544	6	90,66	0,30	0,91
Zone4	538	8	67,25	0,23	0,67
Zone5	1078	10	107,8	0,36	-
Zone6	107	11	9,72	0,03	0,10
Zone7	167	11	15,18	0,05	0,15
Zone8	145	12	12,08	0,04	0,12
Zone9	102	6	17	0,06	0,17
Zone10	112	3	37,33	0,12	0,37
Zone11	209	13	16,08	0,05	0,16
Zone12	263	3	87,66	0,29	0,87

Table 4.6: Loading performances

The results clearly show that if we consider a medium size vehicle for delivering, it needs to travel often with not full load. This is done because of

both the so fragmented delivers typical of *Nanostores* and the aim to minimize the total distance travelled. In fact, no penalty in the objective function is introduced for the vehicles that travels with not full load in our model. Thanks to the information given by the Company, we know also that vehicles with a smaller capacity are available. The table shows the possibility to improve the load factor for the deliveries choosing the vehicles with small capacity. In this case, the load factors increase, but they are far from the condition of full load travelling as the same. Our result can seem a bit surprising, considering that the data suggest the presence also of vehicles with a very big capacity in reality. The fact can be easily explained thanks to the wide different configuration of the *Nanostores*. As explained in *Appendix A*, the term *Nanostores* includes final shops of different dimension, from kiosks to large mini-markets. For easily conducting the analysis, we consider only the medium case, but the necessity to manage so different shops suggest a great level of variance in the deliver quantity that cannot be completely neglected. In a full operational VMI system, the company has to be able also to face these fluctuations that justify also the presence of bigger vehicle capacity. For an in depth overview of the aspects it is possible to conduct other analysis on different vehicles capacity and on the accuracy of the demand prediction in this particular channel.

Considering the possibility to invest for building of a VMI system, it is necessary to understand which typology of vehicle is more suitable for the condition of the environment. *Our results underlines the necessity to consider a fleet composed of* ***small vehicle for the major part****, in order to easily move in complex and huge urban areas, and to better face the possible optimization schedule of a VMI system that manage lots of Nanostores. This is truer if the possibility of a multi-depot scheme is considered.* Also considering the case of an outsourcing for delivery service, it is possible to stipulate contracts that considers small carriers or taking into account other features about vehicle size.

4.6 Conclusion

The results presented in this chapter are relevant from an industrial point of view. We are collaborating with Company's centre for innovation and management of the supply chain, with the aim of formulating and testing optimization models and methods for the application of the VMI systems in complex network with a high presence of *Nanostores*. We investigated, formulated and solved the problem, and we introduced an application on a data set provided by the Company for Surabaya last-mile deliveries. We obtained good results in terms of reduction of the total distances travelled, this evidence underlines the potential saving costs applying the VMI in the last-mile deliveries. We are able to solve big instances up to 138 customers in a real context in acceptable computational time also for the Company point of view. Lots of aspects are suggested to be deeply investigate if the Company maintains its interest on projecting a VMI system in the last-mile. Precisely, it is possible to introduce more constraints in the problem or add more factors in the objective function in order to take into account another trade-off between different types of costs in the optimization. Other improvements could be done trying to correct the final solution with some techniques that re-combine the routes to obtain full load deliveries. Finally, the possibility to use small vehicles without losing the effectiveness in routing opens some perspective in considering a sustainable management of the fleet, introducing green small vehicle and cargo-bikes for the last-mile deliveries in big cities. The situation of Surabaya city is very similar to other big cities, especially in emerging markets, so implementing a VMI system could be a good opportunity for saving money in terms of transportation costs for the last-mile and can also represents a solution with an elevated level of scalability and portability in different markets and urban areas all around the world.

4.7 Acknowledgments

The collaboration with the Company give us the opportunity of reinforcing our knowledge about inventory and routing problems. Valuable advices were obtained from people who daily face challenges in logistics world and business. We particularly thank the two managers that followed the projects for their support and input.

Conclusion

The thesis is focused on the implementation of a VMI setting in a large urban environment and on the study of the Inventory Routing Problems. In particular, the thesis is inspired by applying the VMI paradigm in the last–mile delivery according to the needed of an Industry Company. We focus on two main variants of the IRP: the *Multi-depot IRP* and the *Multi-attribute IRP*. The thesis is divided in four chapters.

In *Chapter 1* we study the *Multi-depot IRP* with homogeneous vehicle fleet and deterministic demand. A basic branch–and–cut algorithm is used to solve the MIP formulation of the problem. The performance of the branch–and–cut are improved by pre–assigning customers to the depots. The clustering method is inspired by the well-known Capacitated Concentrator Location Problem. Classical benchmark instances are solved with this clustering–based branch–and–cut algorithm. The results show that the cluster–based branch–and–cut is always able to find better solutions than the basic branch-and-cut. Clustering methods enhance the standard branch–and–cut algorithm performance to find feasible solutions at the early stage of the search tree. Nevertheless, the main result highlighted in this Chapter is that efficient clustering procedures, in which routing costs are well accounted, represent a power tool to reduce the complexity in solving large $MDIRP$ instances.

In *Chapter 2* we design a more sophisticated matheuristic algorithm to solve $MDIRP$ instances defined over a time horizon of six periods. We propose a clustering formulation based on a mathematical definition of critical customers

around which the urban space is partitioned into clusters. For each cluster and for each pair of clusters, a set of intra and inter cluster routes is built. At the end of this phase, a route-based formulation is solved in order to provide a $MDIRP$ feasible solution. The clustering phase is different from the previous one. It is strongly designed in accordance with the features of the IRP. A quantitative measure of the critical level of each customer is proposed according to the distance from the depots and the stock–out risk. Feasible routes are returned by heuristic procedures based on different replenishing policies and the vehicle capacity. All these ingredients contribute to obtain good results. The matheuristic algorithm overcomes the branch–and–cut algorithm in terms of solutions' quality and computational times. The matheuristic is able to provide solutions for instances with cardinality up to 50 customers. This matheuristic was used to solve the case of study proposed by the Company.

In *Chapter 3*, the *Multi-attribute IRP* is investigated. We propose a MIP mathematical formulation which considers the multi–product case with heterogeneous fleet of vehicles and explicit constraints for the route duration. This configuration contains a lots of features typical of real problems. We introduce a Variable Neighborhood MIP Search ($VMNS$) for solving it. The $VMNS$ is composed by two phases that are executed alternatively: a branch-and-cut phase and a local search phase. A sequence of neighborhoods is designed in order to improve the solution. Some preliminary results are provided. In general, the algorithm is really effective in small and medium size instances, while in the large instances with more than 50 customers and 6 periods there is a high risk of being trapped in local optimal solutions. The other challenge is related to the dimension of the neighborhoods: if they are very small it is possible to not improve to the objective function, while if they are large a long time is required to find a feasible solution within them. We introduced simple neighborhoods, that consider one feature of the problem at a time and only some neighborhoods obtained by restricting the search space according to

given values of more than one feature of the problem. Further improvements could be based on in depth study on the behaviour of the neighborhoods and their effectiveness, as well on the implementation of a diversification technique in order to avoid the stagnation.

In *Chapter 4*, the case of study provided by the company is presented. The analysis is set on Surabaya, a big city of the Isle of Java, that presents all the typical features of a mega–city in an emerging market. It is featured by the large presence of nanostores, that increases the complexity of deliveries in the last-mile. After pre–processing the scenarios, we implement a VMI setting in this context applying the algorithm described in *Chapter 2*, and we compare the results obtained with the matheuristic with the ones coming from the delivery policy used by the Company. The good results obtained underline the potential saving costs that can be achieved by extending the VMI setting to the last-mile deliveries. This represents a significant result for the Company that opens interesting perspective in developing VMI to replenish the nanostores.

Chapter 5

Appendix A: Nanostore features and classification

A **Nanostore is a small shop that sells a variety of products (like food, drug and non-food items) conveniently located near shopper homes, for basic daily purchases or emergency needs**. For *FMCG Company*, a *Nanostore* is a small store located in an urban area featured by an high density of population. These traditional channel stores are independent, family-owned stores (like kiosks, convenience or rural shops), they are visited different times by shoppers during the day/week, for buying one or a limited number of items. They are typical of the emerging markets and so diffused in big/mega cities. *Nanostores* have different characteristics: they usually are independent retailers, family business managed by not so expert people in the field. The shops are usually located in the urban area and present a reduced size, with 15 to 40 squared metres of surface for kiosk, up to a maximum of 1000 squared metres for large mini-markets. Furthermore, they do not use large warehouses, their stock capacity is limited to few quantities of goods, and as a consequence, they present high frequency replenishment and small order sizes. *Nanostores* do not usually give a wide choice to their customers because they sell only one or two brands for every product. They do not have ICT systems to man-

age the products or sharing informations because this investment is considered not necessary in relation to the small quantity of goods and information to be treated. It is possible to individuate different **strengths** and **challenges** in the *Nanostores* ' sector. Some **strengths** that ensure a long life cycle for *Nanostore* in the emerging markets and mega-cities are listed below:

- customers perceive a total cost of purchase as lower, because for the people is very difficult to reach big retailers that are often located outside the urban boundaries;
- customers perceive the shopping activity very rapid in *Nanostore*, this is useful because people can do the daily shopping on their way home from the work, without the necessity of driving;
- *Nanostore* presents lower barrier of entry, it is generally necessary a low capital requirement for opening a store of this type and also easy licensing provisions in lots of countries;
- *Nanostore* is featured by a friendly atmosphere that emphasizes the customer loyalty. It is possible to build a trustworthy relationship between the customer and the owner, that concedes informal credits if it is necessary;
- the increased urbanisation on a global level and the high population density in the cities are potential for growth for the *Nanostore* segment.

Despite of lots of possibilities for growth in the *Nanostore* segment, this type of shops is affected by different challenges:

- as said before, *Nanostores* do not use **ICT tools** for analysing, interpreting and sharing data, so they do not help the process of forecasting and planning deliveries by the supplier;

- *Nanostores* usually operate with a **small availability of cash**, for this reason it is not possible to plan big deliveries if the cash is not available for pay them;
- in relation with their own configuration, *Nanostores* risk the **out-of-stock**, this is a negative point for their business because the loyal customer to the brand can decide to change store for shopping;
- *Nanostores* face **low margins** because they are not able to take advantages from scale-economies and they usually resupply also from wholesalers and distributors which already applied a mark-up on the products;
- *Nanostores* owners prefer to sell **small cases** of products taking into account the needs of their customers, but lots of time only big size products are suitable for the market.

According to the new global channel definition, **the *Nanostore* channel is divided in 8 sub-channels: Kiosk, Open Market, Small Traditional, Medium Traditional, Large Traditional, Convenience Stores, Small Mini-Markets, Large Mini-Markets**.

The smallest section of *Nanostores* is composed by Kiosk, Open Market and Small Traditional. **Open Markets** are located in open areas and organized by authorities, the seller usually paid for participation in it. **Kiosks** are the smallest stores with a fixed structure where the decisions are taken by the owner. **Small Traditional** shops reach up to the 20 m^2 (sale area and storage) and support daily or weekly consumption, but also immediate consumables. Their structure is more complex than the other cases analysed and they sell predominantly food categories.

Figure 5.1: Small Nanostore types: from the left to the right an example of Open Market, Kiosk and Small Traditional

The medium size section of *Nanostores* is made up of **Medium and Large Traditional and Convenience Store**. They are generally centred on selling food categories and sometimes they have also fresh products. In the case of the Convenience Stores we can find also alcohol, impulse categories, cigarettes and snacks. They can be self-service or counter service but typically they present hand-written tags. The difference between them is given by their dimension: for Medium Traditional the surface covered is 20-40 m^2; for the Large Traditional it is around 40-100 m^2 and for the Convenience Store up to 300 m^2.

Figure 5.2: Medium Nanostore types: from the left to the right an example of Medium Traditional, Large Traditional and Convenience Store

The large size section of *Nanostores* is composed by **Small Mini Market and Large Mini Market**. They have usually a consolidate structure with self service and scanners used at checkout, checkout lanes and counters, the

presence of different food categories, fresh products and multiple aisles and they can be aggregate with a chain or independent. Also in this case the main difference is the store size: total surface around 100-500 m^2 for Small Mini Markets and 500-1000 m^2 for Large Mini Markets.

Figure 5.3: Large Nanostore types: on the left Small Mini Market and on the right Large Mini Market

Nanostore is a fast growing and strategic channel, there are 24 million *Nanostores* across the world, serving 5 billion consumers. It represents the 60% of developing markets business and for Company it represents up to 40% of the business in relevant markets. This channel covers mainly 4 regions: LA (Latin America), CEEMEA (Central and Easter Europe Middle East and Africa), Greater China, AAI (Australia, ASEAN ans India). This is the reason because the segments is so relevant and so interesting for the company.

Figure 5.4: Regions covered by Nanostores

Bibliography

[1] Adukyasak, Y. and Cordeau, J-F. and Jans,R.: Formulations and Branch-and-Cut Algorithms for Multivehicle Production and Inventory Routing Problems. INFORMS Journal of Computing, 14, 103–120 (2013)

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

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

[4] Archetti, C. and Boland, N. and Speranza, M.G.: A Matheuristic for the Multivehicle Inventory Routing Problem. INFORMS Journal on Computing, 29(3), 377–387 (2017)

[5] Archetti, C. and Bertazzi, L. and Hertz, A. and Speranza, M.G.: A hybrid heuristic for an inventory routing problem. INFORMS J. Comput. 24(1), 101-116 (2012)

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

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

[8] Arh, D.: Contribution to multiple postmen problems, Phd Dissertation. University of Heidelberg, page 117 (2004).

[9] Barahona, F. and Grötschel, M.: On the cycle polytope of a binary matroid. Journal of combinatorial theory, Series B, Elsevier, 40(1), 40–62 (1986)

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

[11] Bektas, T. and Crainic, T.G. and Van Woensel, T: From managing urban freight to smart city logistics networks. Research Papers 2015, CIRRELT-2015-17

[12] Bell, W.J. and Dalberto, L.M. and Fisher, M.L. and Greenfield, A.J. and Jaikumar, R. and Kedia, P. and Mack, R.G. and Prutzman, P.J.: Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. JSTOR, Interfaces, 13, 4–23 (1983)

[13] Bertazzi, L. and Bosco, A. and Guerriero, F. and Laganá, D.: A stochastic inventory routing problem with stock-out. Elsevier, Transportation Research Part C: Emerging Technologies, 27, 89–107 (2013)

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

[15] Bertazzi, L. and Speranza, M.G.: Continuous and discrete shipping stategies for the single link problem. Transportation Science, 36(3), 314–325 (2002)

[16] Bertazzi, L. and Speranza, M.G.: Matheuristics for Inventory Routing Problems in Hybrid Algorithms for Service, Computing and Manufacturing Systems: Routing, Scheduling and Availability Solutions. J.R. Montoya-Torres and A.A. Juan and L.H. Huatuco and J. Faulin and G.L. Rodriguez-Verjan, editors, IGI Global, 1–14 (2011)

[17] Bertazzi, L. and Speranza, M.G.: Inventory routing problems: an introduction. EURO Journal on Transportation and Logistics, Springer, 1, 307–326 (2012)

[18] Bertazzi, L. and Speranza, M.G.: Inventory routing with multiple customers. EURO Journal on Transportation and Logistics, Springer, 2, 255–275 (2013)

[19] Bertazzi,L. and De Maio, A. and Laganà, D. : The impact of a clustering approach on solving the multi-depot IRP. International Conference on Optimization and Decision Science, Springer, pages 507-515 (2017)

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

[21] Bretzke, W. R.: Global urbanization: A major challenge for logistics. Logistics Research, 6 (2-3),57–62 (2013)

[22] Campbell, A.M. and Savelsbergh, M.: A decomposition approach for the Inventory Routing Problem. Transportation Science, 38(4), 488–502 (2004)

[23] Coelho, L.C. and Laporte, G.: A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. International Journal of Production Research. 2013 Nov 18; 51(23-24):7156-69

[24] Coelho, L.C. and Laporte,G.: The exact solution of several classes of inventory-routing problems. Computers & Operations Research, Volume 40, Issue 2, 2013, Pages 558-565

[25] Coelho, L.C. and Cordeau, J.F. and Laporte, G.: Thirty Years of Inventory–Routing. Transportation Science, 48, 1–19 (2013)

[26] Coelho, L.C. and Laporte, G.: Improved solutions for inventory-routing problems through valid inequalities and input ordering. Internat. Journal of Production Economics, 155, 391–397 (2014)

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

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

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

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

[31] Danna, E. and Rothberg, E. and Pape, C. L.. Exploring relaxation induced neighborhoods to improve MIP solutions. Mathematical Programming, 102(1), 71–90 (2005)

[32] Dauzère-Pérès, S. and Nordli, A. and Olstad, A. and Haugen, K. and Koester, U. and Per Olav, M. and Teistklub, G. and Reistad, A.: Omya Hustadmarmor optimizes its supply chain for delivering calcium carbonate slurry to European paper manufacturers, INFORMS Interfaces, 37(1), 39–51 (2007)

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

[34] M.M. Deza, E. Deza: Encyclopedia of Distances. Springer, Berlin, Heidelberg (2009)

[35] Dillenberger,C. and Escudero, L.F. and Wollensak, A. and Zhang, W. : On practical resource allocation for production planning and scheduling with period overlapping setups. EJOR 75(2), 275–286 (1994)

[36] Dondo, R. and Cerda', J.: A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. European Journal of Operational Research, 176(3), 1478–1507 (2007)

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

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

[39] Ghiani, G. and Laporte, G. and Musmanno, R.: Introduction to Logistics Systems Management. Wiley Series in Operations Research and Management Science (2013)

[40] Gouveia, L. and Saldanha-da-Gama, F.: On the capacitated concentrator location problem: a reformulation by discretization. Computers & Operations Research, 33, 1242–1258 (2006)

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

[42] Guzelsoy, M. and Nemhauser, G. and Savelsbergh, M.: Restrict-and-relax search for 0-1 mixed-integer program, EURO Journal on Computational Optimization, volume 1(1-2), 201–218 (2013)

[43] Larrain, H. and Coelho, L.C. and Catald, A.: A Variable MIP Neighborhood Descent algorithm for managing inventory and distribution of cash in automated teller machines. Computers & Operations Research, 85(C), 22–31 (2017)

[44] Mladenović,N. and Hansen,P.: Variable neighborhood search. Computers & Operations Research, Volume 24, Issue 11, 1997, Pages 1097-1100

[45] Noor, N. M. and Shuib, A.: Multi-Depot Instances for Inventory Routing Problem Using Clustering Techniques. Journal of Industrial and Intelligent Information, 3(2), 97-101, June(2015)

[46] Padberg, M. and Rinaldi, G. : A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. SIAM review, 33(1), 60–100 (1991)

[47] Popović, D. and Vidović, M. and Radivojević, G.: Variable neighborhood

search heuristic for the inventory routing problem in fuel delivery. Elsevier, Expert Systems with Applications, 39(18), 13390–13398 (2012)

[48] Prodhon, C. and Prins, C. : A survey of recent research on location-routing problems. European Journal of Operational Research, 238, 1–17 (2014)

[49] N. Ramkumar, P. Subramanian, T.T. Narendran, and K. Ganesh: Mixed integer linear programming model for multi-commodity multi-depot inventory routing problem. Opsearch, 49(4), 413–429 (2012).

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

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

[52] Solyalı, O. and Cordeau, J.-F. and Laporte, G.: Robust inventory routing under demand uncertainty. INFORMS Transportation Science, 46, 327–340 (2012)

[53] Solyalı, O. and Süral, H.:A single supplier–single retailer system with an order-up-to level inventory policy. Elsevier, Operations Research Letters, 35(5), 543–546 (2008)

[54] Developed by William Cook: Concorde TSP Solver, http://www.math.uwaterloo.ca/tsp/concorde.html (Latest update 2016).

Acknowledgements

Along the road that leads to a desired goal, one is never alone. Therefore I would like to thank the people who guided me during these years: my supervisor Prof. Demetrio Laganà, for his careful guidance and supervision in my growing from the scientific and professional point of view, for his capacity to transmit his passion for the scientific research, for his ability to listen, advise and encourage me to always go beyond my limits; Prof. Leandro Callegari Coelho, for his warm welcome and encouragement to my arrival in Canada, for his contribution to enrich my knowledge and my professional and personal skills; the two managers that guided me during my internship in Brussels, for giving me valuable advices in order to enrich my thesis also from a managerial point of view; Prof. Roberto Musmanno, that believed in me from the beginning and always said to me words of encouragement.

Thanks to all the colleagues and friends at the university, in particular to Giusy, that shared with me this experience from beginning to end, becoming my favorite companion of adventures overseas (and not only); to Carlo, that always gave me careful and thoughtful advices, sharing his experience with me and his tireless support; to all the other collegues (Gianluca, Rosario, Antonio V., Antonio C., Claudio, Sarah, Elisa N., Vittorio, Aurora, Maria, Claudia and the components of the Operations Research Group) that created a stimulating and friendly working environment.

Thanks to all the people who loves me: Mum, Dad, Luisa, Marco, Dany, Isa, Maryam, Arianna, Antonella, Katia, Isabella, Silvia, Vanessa, Federica, Elisa B., Vincenzo, all the IGeA guys and many others...thanks to whom in big or small way gave me his support, affection, encouragement or experience to get to the top of the climb, because I was able to walk more confident with these precious gestures.

Finally, thanks also to who did not believe in me, because it gave me an extra push to get to the end.

Annarita D.