Managing the Inventory and Distribution of Cash in Automated Teller Machines Using a Variable MIP Neighborhood Search Algorithm

Homero Larrain
Leandro C. Coelho
Alejandro Cataldo

January 2016

CIRRELT-2016-01

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2016-001.

Managing the Inventory and Distribution of Cash in Automated Teller Machines Using a Variable MIP Neighborhood Search Algorithm

Homero Larrain[1,*], **Leandro C. Coelho**[2], **Alejandro Cataldo**[3]

[1] Departmento de Ingeniería de Transporte y Logística, Pontificia Universidad Católica de Chile, Avda. Vicuña Mackenna 4860, Macul, Santiago, Chile

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, 2325 de la Terrasse, Université Laval, Québec, Canada G1V 0A6

[3] Departmento de Ingeniería Industrial y de Sistemas, Pontificia Universidad Católica de Chile, Avda. Vicuña Mackenna 4860, Macul, Santiago, Chile

Abstract. In this paper we solve a practical problem arising in the ATM management and replenishments in Santiago, Chile. This rich and challenging problem shares much of its structure with the inventory-routing problem, but some features make it unique. We model the problem and solved it by branch-and-cut. In order to efficiently tackle large instances, we propose a new Variable MIP Neighborhood Search (VMNS) heuristic, which takes advantage of the MIP implementation. The VMNS relies on the mathematical formulation of the problem and significantly simplifies its resolution. We derive neighborhoods to exploit the structure of the problem at hand, be it over routes, locations, periods or quantities delivered. Based on extensive computational experiments, our VMNS is shown to significantly improve the solutions from the exact method using only a fraction of the time. Sensitivity analyses are performed to confirm the robustness and effectiveness of our method.

Keywords. Automated teller machines, optimization, cash industry, neighborhood search.

Acknowledgements. We thank the contact persons at our industrial partner for providing the data and validating our parameters. We thank Calcul Québec for providing high performance parallel computing facilities. This project was partly funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant 2014-05764. This support is greatly acknowledged.

* Corresponding author: homero@ing.uc.cl

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2016

1 Introduction

In this paper we introduce, model and solve an inventory control and distribution management problem arising in the bank service industry, namely managing the amount of cash for automated teller machines (ATMs) in Santiago, Chile. This problem presents all the features of a rich inventory-routing problem (IRP). The IRP combines two well-known problems in a multi-period time horizon: vehicle routing and inventory management. In this problem a decision maker has to determine when to visit each ATM, how much cash to deliver each time, and how to assign ATMs to be visited into different vehicle routes. The objective of the problem is to jointly minimize the total inventory handling and distribution costs, while serving the demand in a satisfactory fashion.

There are many different variations and applications of the IRP. A survey on its practical aspects is provided by Andersson et al. [3] and a classification and an extensive literature review of the problem is given in Coelho et al. [14]. One of the possible variations of this problem occurs when stockouts are allowed. Stockouts can be handled as lost demand as in Aghezzaf et al. [2], or backlog to be satisfied in future periods as in Abdelmaguid and Dessouky [1]. In any case, stockouts are penalized in order to be avoided. This is the context in which our problem is defined. The IRP with stockouts typically appears when stochastic demands are considered explicitly [9, 15], although this is not the case for this work. Different algorithms have been proposed to solve the problem, ranging from local search heuristics [8, 13, 22], exact algorithms [4, 11, 16], and hybrids of heuristics and exact methods [5]. We are aware of only one paper addressing the IRP arising in the banking industry [21], where the authors propose a decomposition scheme to provide solutions for an industrial partner in the Netherlands, where recirculation ATMs are available.

The resolution of very rich and challenging problems such as ours calls for new powerful and hybrid algorithms, i.e., those combining mathematical programming and heuristics [17, 20]. Hybrid methods have been used successfully to solve the green vehicle routing problem [7]. One of such methods [19] employs heuristics in a first phase to construct a set of possible routes to then, in the second phase, use this set of routes to build a feasible and improved solution to the problem by means of a set partitioning formulation over the routes. The same hybrid method has been applied to solve the minimum common string partition and the minimum covering arborescence problems, and the results have shown that the proposed algorithm is a state-of-the-art method for these problems, especially for what concerns rather large problem instances [10]. On a non-linear IRP, as will be our case, Mirzaei and Seifi [18] have shown that the use of heuristics performs better than solving the nonlinear program. Moreover, it is shown that when the size of the input data increases, the optimality gap grows exponentially when using a nonlinear programming solver, but only linearly when using ad-hoc heuristics.

Inspired by real-life data provided by an industrial partner from Santiago, Chile, our problem is significantly different than that of van Anholt et al. [21] in the sense that we have a single depot from which all ATMs must be replenished, yielding an instance much larger than the ones handled by their method. Moreover, our industrial partner explicitly allows stockouts, which are discouraged with a penalty whenever they occur, and our planning horizon must be more finely controlled, with some ATMs requiring visits during a specific period of the day (morning, afternoon or during the night). Furthermore, due to security reasons and the ATM technology employed in Chile, replenishments are performed by switching sealed money containers within the ATMs. This last issue defines an unconventional inventory policy which slightly differs from the ones studied more frequently in the literature, i.e., maximum-level (ML) and order-up-to-level (OU). This feature will be detailed in Section 3. Our partner currently clusters ATMs by small geographical areas and the replenishments are handled by a team of dispatchers.

The objectives of this research are twofold. First, we develop an ad-hoc exact branch-and-cut (B&C) algorithm for the IRP in which stockouts are allowed. Second, in order to cope with the very large instances provided by our partner, we formalize and implement a hybrid approach in which we apply B&C and a series of mixed-integer programming-based heuristics to find good solutions within short computing times. This new flexible algorithm, which we call Variable MIP Neighborhood Search (VMNS) is shown to yield significantly better solutions that the truncated B&C, using only a fraction of the running time. One of the main advantages of the VMNS is that it relies on the previous implementation of the B&C, making it possible to be easily generalized for other problems for which the user has implemented such a framework. A previous and simplified version of such structure to improve existing solutions has been exploited by Archetti et al. [6] and Coelho and Laporte [12].

The remainder of this paper is organized as follows. In Section 2 we formally define the problem. A mathematical formulation is proposed in Section 3. Our hybrid VMNS solution algorithm is described in Section 4. Section 5 presents extensive computational experiments on real-life based instances, followed by conclusions in Section 6.

2 Problem description

The problem we study in this paper is an IRP variant arising in the ATM replenishing problem faced by our industrial partner in the city of Santiago, Chile. In this problem, a set of ATMs spread throughout the city has to be visited to ensure cash availability to fulfill the demand predictions during a time horizon. The decision maker has to determine when to visit each ATM, how much cash to deliver each time, and which routes to use to deliver the cash. Unlike in the classic IRP, stockouts are allowed but penalized. Therefore, the decision maker's objective is to find a distribution plan minimizing the total of distribution costs, inventory holding costs and stockout penalties, subject to capacity constraints.

In this problem, capacity constraints appear in three forms. First, due to insurance reasons, vehicles are not allowed to carry more than a certain amount of money, imposing a tight capacity issue. Second, routes should not exceed the duration of a period, due to drivers regulation constraints. Finally, ATMs also have a maximum holding capacity. The capacity and holding costs for the depot are not relevant for our problem, but they can easily be considered in the model and solution procedure.

The main features distinguishing our problem from conventional IRPs are:

1. Stockouts are allowed: due to the difficult nature of the problem in practice, some tolerance is given to stockouts, which are allowed to happen, but are penalized. The

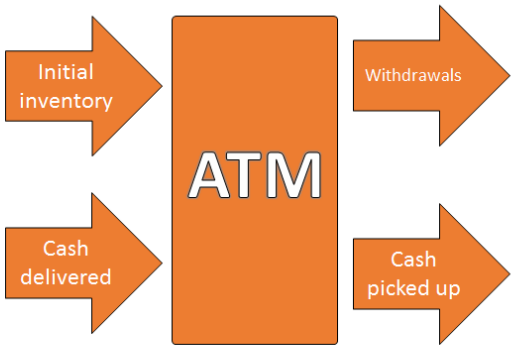
demand occurring during the time an ATM is out of stock is lost, i.e., there is no backlogging.

2. ATMs are replenished using cassettes: the replenishment is performed using sealed cash containers, called cassettes. When an ATM is replenished, the old cassette is removed with all of its contents and replaced with a new cassette, which is always full. This is similar to the OU policy, but not identical.

3. There is a limited number of possible cassette sizes: for security reasons related to money handling at the depot, there is a limited number of cassette sizes (in terms of amount of cash) that can be shipped to an ATM.

4. Some ATMs can only be visited during certain periods: for operational reasons, some ATMs have to be visited in specific periods of the day. For instance, subway stations in Santiago only allow cash replenishment operations during the night shift, while some other establishments, such as grocery stores, can only be accessed during daytime.

Our proposed model solves a deterministic instance of the problem, for which the planner relies on accurate information on demand forecasts, as well as travel and operational times. It is also assumed that replenishments occur at the beginning of each period, regardless of the position of the ATM on its route. The replenishing at the depot is assumed to happen after the vehicles have departed. The logic underlying the dynamics of inventories for a given ATM is illustrated in Figure 1 which shows, for any given period, the sources of cash input and output. Cash in an ATM on a period can come from remaining cash from the previous period, or from a cash delivery during the same period. Cash leaves the ATM in two ways: withdrawals made by the customers, or pick ups made by the operator (when the old cassette is removed and taken back to the depot). These four possible movements can be added up to obtain the net cash balance for an ATM in a given period. If this balance is non-negative, there is no stockout during that period, and the initial inventory for the next period will correspond to this net balance. If the net balance is negative, it means that there is a cash stockout of the same magnitude as the net balance, and that the initial inventory on the next period is null.



Net balance*	≥ 0	< 0
Stock out	0	Net balance
Initial inventory for next period	-(Net balance)	0

*Net balance = Initial inventory + Cash delivered – Withdrawals – Cash picked up

Figure 1: Inventory conservation and cash flow for an ATM

As it often happens in big cities around the world, ATMs in Santiago are usually grouped in some locations. In practice, it is our partner's policy to replenish all the grouped ATMs when visiting a location. Given that we assume demands to be deterministic, if initial conditions for a group of ATMs are the same, and we choose the same cassette sizes for every visit, then every ATM at the same location will always have the same inventory level. This is the current practice of our partner, therefore, we can model a group of ATMs sharing location as a single high-capacity ATM. Depending on the nature of the input data, this assumption can greatly reduce the number of nodes in our network.

Travel times are modeled as the sum of two components: an arc-dependent and node-dependent part. The arc-dependent part is the time it takes to get from one location to the next one (including parking maneuvers and walking), which is assumed to be symmetric. The node-dependent part corresponds to the time it takes to replenish the ATMs in a location. This time is modeled as being directly proportional to the number of ATMs in the location, given that all of them will be replenished at the same time.

As in other IRPs, we assume that replenishments occur at the beginning of each period. In our scenario, this may translate into some stockouts being underestimated. This situation could be prevented by imposing a safety inventory level at ATMs, or by assuming that deliveries take place at the end of the period. In both cases, however, stockouts would be overestimated. In order to precisely describe the dynamics of the system to avoid such inaccuracies, one would need to resort to solutions which would largely increase the size and complexity of the problem. One of these possibilities would be to discretize the periods in small time intervals or to consider continuous time, with a much finer control of the timing of decisions and cost of actions, but at the expense of significantly complicating the model and the solution procedure.

Due to cash handling security issues when preparing the cassettes, vehicles are only allowed to make a single route during a period. Also for practical reasons, ATMs can only be visited once during a period. Unlike some systems like the one in the Netherlands [21], ATMs in Santiago do not allow transshipment of money, i.e., cash being transferred from one ATM to another, nor deposits are allowed. Finally, the denominations in the cassettes are standardized for each cassette size and do not have an impact in our formulation.

3 Mathematical formulation

In our problem, an operator is in charge of replenishing the cash on a set $\mathcal{V}'$ of n locations located throughout a city. Each location hosts a group of a_i ATMs, with $i \in \mathcal{V}'$. The depot, where vehicles start and finish their trips, is denoted as node 0, and the complete set of nodes (i.e., locations) of the problem is denoted as $\mathcal{V} = \mathcal{V}' \cup \{0\}$. The set of arcs, i.e., ordered pairs of nodes (i, j) with $i < j$ of the problem is denoted by $\mathcal{A}$, with $|\mathcal{A}| = m$. A good and reliable demand forecast r_i^t is known for a representative ATM in each location $i \in \mathcal{V}'$, for each period t in a time horizon $\mathcal{T} = \{1, ..., H\}$. The the amount of cash flowing from an external source into the depot in each period $t \in \mathcal{T}$ is denoted by r_0^t. The amount of cash in any ATM in location $i \in \mathcal{V}'$ cannot exceed its maximum capacity, U_i. The operator works with a fleet of $k \in \mathcal{K} = \{1, ..., K\}$ heterogeneous vehicles which can make at most one single trip per period, serving a subset of $\mathcal{V}'$. The capacity of each vehicle in terms of cash is κ_k. ATMs in a location are replenished by at most one vehicle per period, by changing their cassettes. These cassettes come in a finite number of predefined formats. We denote these different cassette sizes as γ_w, with $w \in \mathcal{W} = \{1, ..., W\}$. We define c_{ij} and τ_{ij} as the cost and the time of traveling over any arc $(i, j) \in \mathcal{A}$. The travel time also considers any fixed time involved in getting ready for replacing cassettes at the ATMs. The time it takes to replace the cassette for each ATM is denoted by μ. A route can not exceed θ units of time. We define h_i as the inventory holding cost for each ATM in location $i \in \mathcal{V}'$; and s_i as the cost incurred for the demand lost at each ATM in $i \in \mathcal{V}'$. Finally, a binary parameter δ_i^t takes a value of one if it is possible to visit location $i \in \mathcal{V}'$ in period $t \in \mathcal{T}$, and zero otherwise.

The problem can be stated as a mathematical programming model as follows. We define the integer decision variables x_{ij}^{kt} (with $i < j$) which will take a value of zero when no trip is made by vehicle $k \in \mathcal{K}$ during $t \in \mathcal{T}$, of one when there is a single direct trip between nodes $i, j \in \mathcal{V}$, and of two when there is a round trip between $i, j \in \mathcal{V}$ (this is only allowed when $i = 0$). Binary variables y_i^{kt} take a value of one when node $i \in \mathcal{V}$ is part of the route of vehicle $k \in \mathcal{K}$ during $t \in \mathcal{T}$. Binary variables z_i^t are used to indicate if there is a stockout occurring in the ATMs from location $i \in \mathcal{V}'$ during $t \in \mathcal{T}$. Continuous variables q_i^{kt} and p_i^{kt} represent the amount of cash delivered and picked up in each ATM from location $i \in \mathcal{V}'$ by vehicle $k \in \mathcal{K}$ in period $t \in \mathcal{T}$, respectively. The inventories (at the beginning of each period) and stockouts for each ATM are modeled by the continuous variables I_i^t and S_i^t, with $i \in \mathcal{V}$ and $t \in \mathcal{T} \cup \{H+1\}$. Initial inventories, I_i^0, are known and fixed for every node $i \in \mathcal{V}$.

The problem can be formulated as follows:

$$\text{minimize} \sum_{i \in \mathcal{V}'} \sum_{t \in \mathcal{T} \cup \{H+1\}} (a_i h_i I_i^t + a_i s_i S_i^t) + \sum_{(i,j) \in \mathcal{A}} \sum_{k \in \mathcal{K}} \sum_{t \in \mathcal{T}} c_{ij} x_{ij}^{kt}, \tag{1}$$

subject to

$$I_0^t = I_0^{t-1} + r^{t-1} - \sum_{i \in \mathcal{V}'} \sum_{k \in \mathcal{K}} a_i q_i^{kt-1} + \sum_{i \in \mathcal{V}'} \sum_{k \in \mathcal{K}} a_i p_i^{kt-1} \quad t \in \mathcal{T} \cup \{H+1\} \tag{2}$$

$$I_i^t - r_i^t + \sum_{k \in \mathcal{K}} q_i^{kt} - \sum_{k \in \mathcal{K}} p_i^{kt} \leq U_i(1 - z_i^t) \quad i \in \mathcal{V}' \quad t \in \mathcal{T} \tag{3}$$

$$-I_i^t + r_i^t - \sum_{k \in \mathcal{K}} q_i^{kt} + \sum_{k \in \mathcal{K}} p_i^{kt} \leq U_i z_i^t \quad i \in \mathcal{V}' \quad t \in \mathcal{T} \tag{4}$$

$$I_i^t = \max\{I_i^{t-1} - r_i^{t-1} + \sum_{k \in \mathcal{K}} q_i^{kt-1} - \sum_{k \in \mathcal{K}} p_i^{kt-1}, 0\} \quad i \in \mathcal{V}' \quad t \in \mathcal{T} \cup \{H+1\} \tag{5}$$

$$S_i^t = \max\{-I_i^{t-1} + r_i^{t-1} - \sum_{k\in\mathcal{K}} q_i^{kt-1} + \sum_{k\in\mathcal{K}} p_i^{kt-1}, 0\} \quad i \in \mathcal{V}' \quad t \in \mathcal{T} \cup \{H+1\} \tag{6}$$

$$I_i^t + \sum_{k\in\mathcal{K}} q_i^{kt} - \sum_{k\in\mathcal{K}} p_i^{kt} \le U_i \quad i \in \mathcal{V}' \quad t \in \mathcal{T} \tag{7}$$

$$p_i^{kt} = y_i^{kt} I_i^t \quad i \in \mathcal{V}' \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{8}$$

$$q_i^{kt} = \sum_{w\in\mathcal{W}} \gamma_w v_i^{ktw} \quad i \in \mathcal{V} \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{9}$$

$$\sum_{w\in\mathcal{W}} v_i^{ktw} = y_i^{kt} \quad i \in \mathcal{V} \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{10}$$

$$\sum_{k\in\mathcal{K}} y_i^{kt} \le 1 \quad i \in \mathcal{V}' \quad t \in \mathcal{T} \tag{11}$$

$$q_i^{kt} \le U_i y_i^{kt} \quad i \in \mathcal{V}' \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{12}$$

$$x_{ij}^{kt} \le y_i^{kt} \quad (i,j) \in \mathcal{A} \quad i \neq 0 \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{13}$$

$$x_{0i}^{kt} \le 2y_i^{kt} \quad i \in \mathcal{V} \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{14}$$

$$\sum_{i\in\mathcal{V}'} a_i q_i^{kt} \le \kappa_k y_0^{kt} \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{15}$$

$$y_i^{kt} \le \delta_i^t \quad i \in \mathcal{V}' \quad t \in \mathcal{T} \tag{16}$$

$$\sum_{(i,j)\in\mathcal{A}} \tau_{ij} x_{ij}^{kt} + \sum_{i\in\mathcal{V}} \mu y_i^{kt} a_i \le \theta \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{17}$$

$$\sum_{j\in\mathcal{V}, i<j} x_{ij}^{kt} + \sum_{j\in\mathcal{V}, j<i} x_{ji}^{kt} = 2y_i^{kt} \quad i \in \mathcal{V} \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{18}$$

$$\sum_{i\in\mathscr{S}} \sum_{j\in\mathscr{S}, i<j} x_{ij}^{kt} \le \sum_{i\in\mathscr{S}} y_i^{kt} - y_m^{kt} \quad \mathscr{S} \subseteq \mathcal{V}' \quad k \in \mathcal{K} \quad t \in \mathcal{T} \quad m \in \mathscr{S} \tag{19}$$

$$q_i^{kt}, p_i^{kt}, I_i^t, S_i^t \ge 0 \quad i \in \mathcal{V}' \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{20}$$

$$x_{0i}^{kt} \in \{0,1,2\} \quad i \in \mathcal{V}' \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{21}$$

$$x_{ij}^{kt} \in \{0,1\} \quad (i,j) \in \mathcal{A} \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{22}$$

$$y_i^{kt} \in \{0,1\} \quad i \in \mathcal{V} \quad k \in \mathcal{K} \quad t \in \mathcal{T}. \tag{23}$$

The objective function (1) minimizes the total of inventory holding, stockout penalties, and routing costs. Constraints (2) impose the conservation of inventory flows in the depot, while constraints (3) and (4) force z_i^t to take a value of one when a stockout happens. Constraints (5) and (6) ensure the conservation of inventory at every ATM, taking stockouts into account, as illustrated in Figure 1. Constraints (7) ensure that capacity on each ATM is not exceeded at the beginning of the period. Constraints (8) ensure that at the time of replenishment the old cassette is picked up with all its contents. Constraints (9) and (10) guarantee that the quantity to deliver to each ATM matches one of the cassette sizes, and that only one cassette is used per delivery. Constraints (11) impose that each ATM can be visited by at most one vehicle per period, while (12) links the quantities delivered to the routing variables, allowing a vehicle to deliver products to an ATM only if it is being visited. Constraints (13) and (14) link variables x_{ij}^{kt} and y_i^{ky}, ensuring that and arc can only belong to a route if both of its extreme nodes are also visited during the period. Constraints (15) impose the maximum capacity of the vehicles, while constraints (16) ensure that locations are only visited in the periods allowed. Constraints (17) state that the duration of a route cannot exceed the maximum route length. The route duration is computed as the total of travel times between locations and the variable operation time for the ATMs in each visited location. Constraints (18) and (19) are the degree and sub-tour elimination constraints, respectively. Constraints (20) to (23) impose the non-negativity and integrality conditions on the variables of the problem.

The model just presented shows non-linearities in constraints (5), (6) and (8). These three sets of constraints can be linearized as follows. Inventory flow constraints (5) can be replaced by the following set of linear constraints:

$$I_i^{t-1} - r_i^{t-1} + \sum_{k\in\mathcal{K}} q_i^{kt-1} - \sum_{k\in\mathcal{K}} p_i^{kt-1} \le I_i^t \quad i \in \mathcal{V}' \quad t \in \mathcal{T} \cup \{H+1\} \tag{24}$$

$$I_i^t \le I_i^{t-1} - r_i^{t-1} + \sum_{k\in\mathcal{K}} q_i^{kt-1} - \sum_{k\in\mathcal{K}} p_i^{kt-1} + Mz_i^{t-1} \quad i \in \mathcal{V}' \quad t \in \mathcal{T} \cup \{H+1\} \tag{25}$$

$$I_i^{t+1} \le U_i(1 - z_i^t) \quad i \in \mathcal{V}' \quad t \in \mathcal{T}, \tag{26}$$

where M is a big number.

In a similar fashion, stockout constraints (6) can be replaced by the following set of linear constraints:

$$S_i^t \leq -I_i^{t-1} + r_i^{t-1} - \sum_{k \in \mathcal{K}} q_i^{kt-1} + \sum_{k \in \mathcal{K}} p_i^{kt-1} \quad i \in \mathcal{V}' \quad t \in \mathcal{T} \cup \{H+1\}. \tag{27}$$

Finally, constraints (8) can be replaced for the following constraints:

$$I_i^t - U_i(1 - y_i^{kt}) \leq p_i^{kt} \leq U_i y_i^{kt} \quad i \in \mathcal{V}' \quad t \in \mathcal{T} \tag{28}$$

$$p_i^{kt} \leq I_i^t \quad i \in \mathcal{V}' \quad t \in \mathcal{T}. \tag{29}$$

This gives rise a MILP formulation for our IRP. The formulation can be further tightened by adding the following valid inequalities and symmetry breaking constraints, inspired by the ones used in Archetti (2007) and Coelho and Laporte (2013, 2014):

$$x_{ij}^{kt} \leq y_i^{kt} \quad (i,j) \in \mathcal{A} \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{30}$$

$$x_{0i}^{kt} \leq 2y_i^{kt} \quad i \in \mathcal{V}' \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{31}$$

$$y_i^{kt} \leq y_0^{kt} \quad i \in \mathcal{V}' \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{32}$$

$$y_0^{kt} \leq y_0^{k-1,t} \quad i \in \mathcal{V}' \quad k \in \mathcal{K} \backslash \{1\} \quad t \in \mathcal{T} \tag{33}$$

$$y_i^{kt} \leq \sum_{j \in \mathcal{V}', j < i} y_j^{k-1,t} \quad i \in \mathcal{V}' \quad k \in \mathcal{K} \backslash \{1\} \quad t \in \mathcal{T}. \tag{34}$$

Constraints (30) and (31) impose that arcs belonging to routes must visit these ATMs. Costraints (32) impose that ATMs can belong to a route only if that vehicle leaves the depot. Symmetry breaking constraints (33) ensure that the routes are activated in a solution with an increasing vehicle index. Finally, symmetry breaking constraints (34) prevent routes for being interchangeable in a given period.

4 Solution algorithms

In this section we describe the two algorithms used to solve the problem at hand. In section 4.1 we outline the traditional B&C scheme used. In section 4.2 we describe in detail the idea of VMNS and how we have implemented it to solve the IRP for ATMs.

4.1 Branch-and-cut

We now sketch the B&C scheme we have implemented to solve the model, which is largely standard in distribution problems. At any given node of the search tree, the linear program defining the problem, without the sub-tour elimination constraints, is solved. At every node we perform a search for violated sub-tour elimination constraints (19), and some of these constraints are added to the current node which is then reoptimized. This process is repeated until a solution is found, or until there are no more cuts to be added. At this point branching on a fractional variable occurs in a typical branch-and-bound fashion.

4.2 Matheuristic approach: Variable MIP Neighborhood Search

The performance of the B&C algorithm can be improved by the use of local searches whenever it finds a new feasible solution. In our VMNS algorithm, local searches are implemented in such a way as to take advantage of the B&C implementation, based on the notion that a local search problem can be modeled as a constrained subproblem which is considerably easier to solve than the original problem.

In what follows we present in detail the VMNS algorithm. In Section 4.2.1 we describe it in a generic way, using the IRP for ATMs to provide examples, without loss of generality. Then, in Section 4.2.2 we detail the implementation of the algorithm for the problem at hand.

4.2.1 Description of VMNS

4.2.1.1 Overview and definitions

Consider the neighborhood conformed by any solution that is attainable by changing one route and its quantities, without changing anything else, starting from a given solution. For a given reference route, which defines an instance of the neighborhood, we can define an associated local search subproblem that looks for the best solution of that instance of the neighborhood. To generalize this idea, taking advantage of the mathematical formulation, let $\bar{x}_{ij}^{kt}$ and $\bar{q}_i^{kt}$ be the values of the routing variables x_{ij}^{kt} and the delivered quantities q_i^{kt} in the current solution. Performing a local search in this neighborhood for the given instance is the same as solving a constrained version of the original model, by imposing the new constraints $x_{ij}^{kt} = \bar{x}_{ij}^{kt}$ and $q_i^{kt} = \bar{q}_i^{kt}$ for all vehicles k and periods t except for the vehicle and period corresponding to the reference route. Note that since any feasible solution of our problem can be characterized by the values of x_{ij}^{kt} and q_i^{kt} (i.e., x_{ij}^{kt} and q_i^{kt} are the independent variables of the problem), it is not necessary to include the other variables from the problem (i.e., inventories, pick-ups, etc.) when defining local search subproblems.

Following the nomenclature just introduced, a particular neighborhood or subproblem is seen as a collection of instances. When performing a local search as a constrained subproblem, choosing a small neighborhood will make its instances easier and faster to solve. However, a smaller domain also means a smaller chance to find a better solution in it. So, whenever the algorithm has to perform a local search around a solution, VMNS first tries with the smallest neighborhoods, and move on to a larger neighborhood when all of its instances are exhausted.

Our algorithm is described schematically in Figure 2. In this context, exploring a neighborhood means to solve its different instances until finding a new upper bound for the problem. On a broad scale, VMNS works by iterating between the local search phase and the B&C phase. The time that the algorithm spends solving the B&C problem is set as a function of the time spent on the last local search phase, in order to maintain a balance between the two phases.

Before entering these two phases, an initial solution is required to start the algorithm. The trivial solution where no routing is made is feasible to our problem. This solution, however, will typically be rather inefficient, yielding very high stockout costs.

4.2.1.2 Exploring a neighborhood

The local search stage can be seen as the exploration of a series of predefined neighborhoods $\omega \in \Omega$ in order of (approximate) increasing size or difficulty. Exploring a

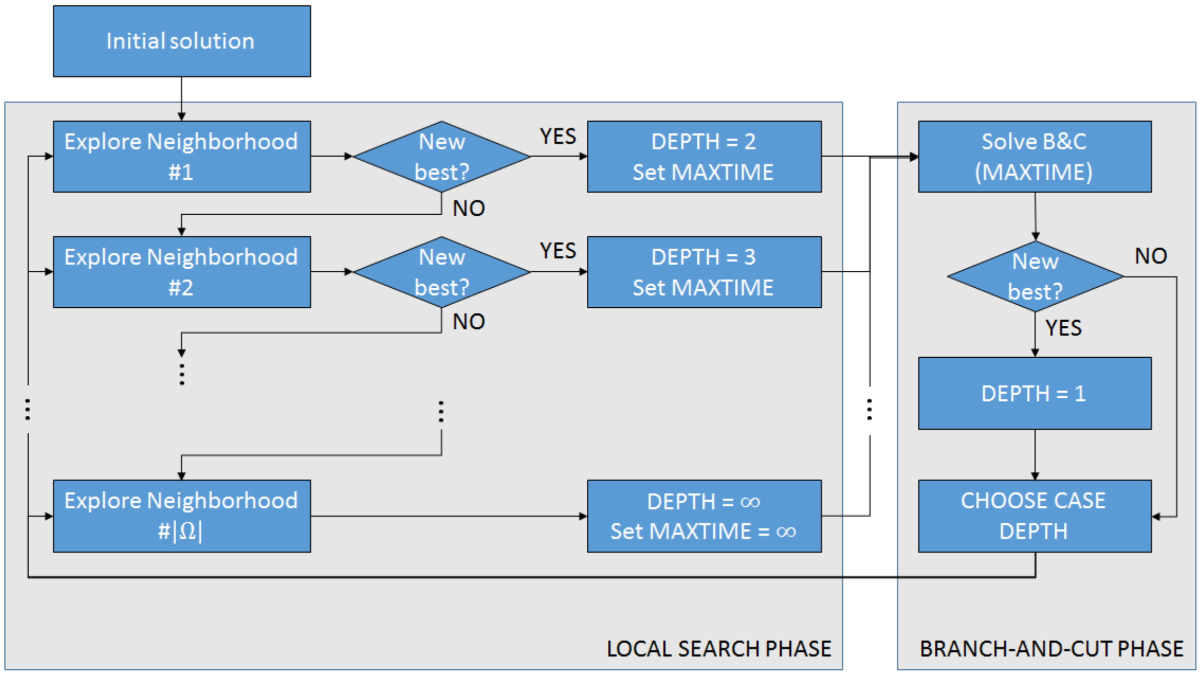


Figure 2: Diagram of VMNS matheuristic procedure

neighborhood involves exploring all of its instances repeatedly until no improvement can be found. Every instance is formulated as a constrained subproblem of the original model, where a subset of the variables is fixed to the value it takes in the current best known solution, and the remaining variables are free. These instances are solved using the same B&C algorithm we implemented for the original problem, which can be easily adapted to take into account the new constraints as lower and upper bounds.

For the example neighborhood mentioned earlier, i.e, “change one single route” or “routes” (for short), an instance is defined by a vehicle $k \in \mathcal{K}$ and a period $t \in \mathcal{T}$ (i.e., “change the route performed by vehicle k in period t”), and the set of free variables would correspond to x_{ij}^{kt}, $\forall (i,j) \in \mathcal{A}$. The dependent variables of the problem are always left free to avoid infeasibilities. To explore the “routes” neighborhood we will need to loop through the instances defined by the different possible values of k and t; we will refer to these as the looping indexes of the neighborhood. We also define the set $FIX_{\omega}(LoopingIndexes)$ as the set of variables x_{ij}^{kt} that remain fixed when exploring an instance of neighborhood ω defined by certain values of the looping indexes.

A pseudocode for a generic version of the neighborhood exploration subproblems solved during the local search phase is presented in Algorithm 1. To understand the pseudocode, the instructions “fix”, “free”, “increment” and “initialize” need to be defined. In this context, fixing a variable means to set both its lower and upper bounds to the current value of the variable from the best known solution. Freeing a variable means setting the lower bound to zero and the upper bound to the corresponding upper bound from the original problem. Incrementing the looping indexes means to move to a new combination of indexes, cycling through their feasible values for a neighborhood. For example, in the “routes” neighborhood, the indexes (k, t) go from $(1, 1)$ to (K, H) and then back to $(1, 1)$, covering all the possible combinations. Finally, initializing the indexes means setting them to any feasible initial value.

4.2.1.3 Switching between phases

The different neighborhoods are explored sequentially from smallest to biggest until one of them yields a new best solution. When this happens, the algorithm exits the local search phase. At this point, the algorithm sets a maximum time for the B&C phase. If this time is exhausted without finding an improvement, the algorithm re-enters the local search phase to try a bigger neighborhood, because the current solution could not be improved in the smaller ones. If a new solution is found by the B&C within the allotted time, the algorithm switches to the local search phase, starting again from the smallest neighborhood.

There are several possible criteria to set the maximum time to be allotted to the B&C phase. In our implementation, we set it as α times the time spent in the local search stage, unless the largest neighborhood has been reached. If a new solution is found in the largest neighborhood, or if no new best solution is found during the local search stage, it is useless to come back to the local search stage unless a new solution is found in the B&C phase. Thus, in these cases the algorithm sets the maximum B&C time to infinity and it only switches back to the local search phase when a new best solution is obtained.

The depth parameter d is defined to store the neighborhood where the local search should start exploring when the B&C phase is over. Whenever exiting the local search phase this parameter is set as the next unexplored neighborhood. If there is no unexplored neighborhood left, the algorithm sets the depth to infinity. When exiting the B&C phase, if a new solution was found, the depth parameter is updated to take the value of the smallest neighborhood, and no updating is made otherwise. This is illustrated in Figure 2.

The B&C phase starts by invoking the B&C algorithm, providing it with the new best solution if one was found. The solution time is limited to the maximum time defined in the previous stage. To take advantage of the simplicity and improving potential of the local search problems, the B&C stage stops as soon as a new best solution is found.

Algorithm 1 Generic neighborhood exploration algorithm

```
1: Initialize $CurrentLoopingIndexes$.
2: $ExitLoopingIndexes \leftarrow CurrentLoopingIndexes$.
3: Set $InitialCost$ and $CurrentBestCost$ as the cost of the current solution for the problem.
4: repeat
5:     Fix $x_{ij}^{kt}$ in $FIX_{\omega}(CurrentLoopingIndexes)$, where $\omega$ is the current neighborhood.
6:     Solve the model using B&C and obtain $NewCost$.
7:     if $NewCost < CurrentBestCost$ then
8:         $CurrentBestCost \leftarrow NewCost$.
9:         $ExitLoopingIndexes \leftarrow CurrentLoopingIndexes$.
10:        Increment $CurrentLoopingIndexes$ for neighborhood $\omega$.
11:    end if
12: until $ExitLoopingIndexes = CurrentLoopingIndexes$.
13: $Improvement = InitialCost - CurrentBestCost$.
14: return $Improvement$.
```

4.2.2 Application of VMNS to the IRP for ATMs

In the implementation of the algorithm we report in this paper, we have developed five different neighborhoods. The first one is defined as changing the quantities (i.e., the cassette sizes) for every location and period, without modifying the routes. The second neighborhood is defined as changing one route at a time (like the one used as an example earlier). The third one is defined as changing all the routes belonging to a period. The fourth one consists in changing all the routes across periods for one specific vehicle. Finally, the fifth neighborhood consists in changing the routes from two periods, fixing the rest. Since in the context of our problem variables q_i^{kt} add little complexity to the subproblems, we have decided to keep all of them free in every scenario. Table 1 presents these five neighborhoods and their defining attributes. Note that the looping indexes in the "two periods" neighborhood are defined in a way that $t_1 < t_2$ so no duplication is made. For the free variables, we use the bullet symbol "$\bullet$" meaning that an index can take any value, i.e., $x_{\bullet\bullet}^{kt}$ is equivalent to $x_{ij}^{kt}, \forall (i,j) \in A$. Note that for each neighborhood the set FIX_ω can be obtained as the set of variables x_{ij}^{kt} which are not free.

As mentioned earlier, the idea behind the algorithm is to perform local searches starting from the smaller, easier neighborhoods. Neighborhood difficulty will naturally depend on the instance size and the number of instances to solve. However, since the complexity of routing problems grow exponentially with size, instance size is much more critical for neighborhood difficulty. We have also derived our neighborhoods and their sequence to avoid redundant searches.

Following this criteria, we have decided to rank the neighborhoods approximately by instance size, leaving "Vehicles" between "Periods" and "Two Periods" because these last two are similar in nature. Figure 3 illustrates the neighborhoods, showing in thicker lines the elements corresponding to fixed variables (arcs for x_{ij}^{kt}, nodes for q_i^{kt}).

5 Computational experiments

In this section we provide extensive computational experiments to assess the performance of our algorithms and to solve the practical problem faced by our industrial partner. In Section 5.1 we describe the data and the scenarios we have used to evaluate our algorithms, and in Section 5.2 we describe the main results obtained. Detailed sensitivity analyses

Table 1: Neighborhood definitions for the problem

ω	Name	Looping indexes	Free variables*	Instance size†	Number of instances
1	Quantities	–	$q_{\bullet}^{\bullet\bullet}$	1	1
2	Routes	$k = 1, ..., K,\ t = 1, ..., H$	$x_{\bullet\bullet}^{kt},\ q_{\bullet}^{\bullet\bullet}$	m	$K \times H$
3	Periods	$t = 1, ..., H$	$x_{\bullet\bullet}^{\bullet t},\ q_{\bullet}^{\bullet\bullet}$	$m \times K$	H
4	Vehicles	$k = 1, ..., K$	$x_{\bullet\bullet}^{k\bullet},\ q_{\bullet}^{\bullet\bullet}$	$m \times H$	K
5	Two Periods	$t_1 = 1, ..., H,\ t_2 = t_1 + 1, ..., H$	$x_{\bullet\bullet}^{\bullet t_1},\ x_{\bullet\bullet}^{\bullet t_2},\ q_{\bullet}^{\bullet\bullet}$	$2 \times m \times K$	$\frac{1}{2} \times H \times (H - 1)$

*Every x_{ij}^{kt} not belonging to the set of free variables will belong to Fix_{ω}.

†Instance size is measured as an upper bound for the number of free x_{ij}^{kt} variables.

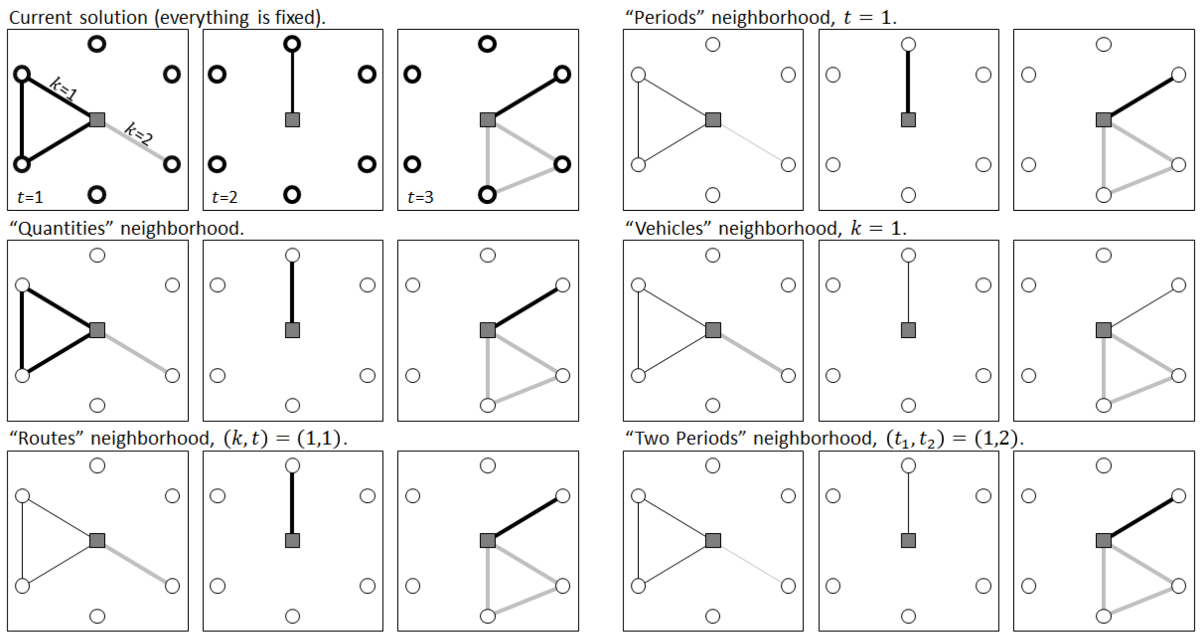

Current solution (everything is fixed).
k=1
k=2
t=1
t=2
t=3
"Quantities" neighborhood.
"Routes" neighborhood, $(k, t) = (1,1)$.
"Periods" neighborhood, $t = 1$.
"Vehicles" neighborhood, $k = 1$.
"Two Periods" neighborhood, $(t_1, t_2) = (1,2)$.

Figure 3: Neighborhood definitions

are performed in Section 5.3.

5.1 Data and scenarios

The algorithms described in Section 4 were tested in scenarios generated based in data provided by our industry partner in the city of Santiago, Chile. From this database we took a random sample of 60 locations yielding instances that are larger than those currently managed manually by the dispatchers of the company. For each location, the database includes geographical data, information on the period-related replenishment restrictions for each location, the expected daily demand for a representative ATM in the location, R_i, and the number of ATMs in the location. Note that both geographical and monetary units have been adjusted to preserve the confidentiality of our partner's data. Monetary values will be represented using the dollar sign, but this does not mean that cash amounts are measured in dollars. The following procedures have been developed in close collaboration with our partner to ensure a true representation of their situation and practice.

In order to generate the demand of a representative ATM for each location in each period, we assume that the daily demand is a random variable following a triangular distribution bounded between $0.5R_{it}$ and $1.5R_{it}$, with mode R_{it}. A day is divided into three time periods (morning, afternoon and night), corresponding to the shifts used by our partner in practice. The total daily demand R_i can be decomposed into the expected demand for each period, r_{it}. Advised by our partner, we assumed that morning and afternoon periods capture 45% of the demand each, and that night periods capture 10%. Following this criteria we have generated three different demand instances, with a 18 period (six days) time horizon. Initial inventories were generated for each instance as a uniform random variable between 0 and U_i.

As for cassette sizes, our partner suggested that we worked with two possible values, 40$ and 65$. Each ATM is assumed to have a maximum capacity of 65$. Vehicles capacities are set at 300$, according to the limits imposed by insurance contracts, and routes are limited to 4 hours of length.

The traveling cost for each arc c_{ij} is computed from the euclidean distances between nodes, using an operational cost of 6\$/km. Travel times τ_{ij} are computed from the euclidean distance as well, using a speed of 20 km/h, adding a fixed time of 10 min for parking the vehicle and accessing the ATMs. The variable time per ATM to replenish is set as 4 min. Holding costs at ATMs are set at 0.5\$ per unit of cash per period, and stockout costs are set at 30\$. All these values were validated by our partner.

To test the performance of the B&C algorithm and of the VMNS matheuristic we have generated scenarios with different network sizes, planning horizons, and demand values. We have generated three different demand scenarios for a network of 60 locations, for an horizon of 18 periods. Then, from these three demand scenarios we have built 27 scenarios, considering the first 20, 40 or 60 locations (as depicted in Figure 4), and the first six, 12 or 18 periods. The fleet size was selected for each instance according to the number of locations: two vehicles for the 20 location scenarios, and three vehicles for the larger cases. We will refer to any particular scenario using the notation $n/H/DemandScenario$, where the first two parameters are the number of locations and periods, and the last one can take values 1, 2, or 3 to refer to a specific demand scenario. These instances can be made available upon request.

5.2 Computational results

The 27 scenarios described in the previous section were optimized using the methods described in Section 4, implemented in C++, and using IBM Concert Technology and CPLEX 12.6. All computations were executed on a grid of Intel Xeon™ processors running at 2.66 GHz with up to 48 GB of RAM installed per node, with the Scientific Linux 6.1 operating system.

Table 2 shows the results for the B&C algorithm on the 27 scenarios. Execution time was limited to three hours. It can be observed that although the exact method is able to find optimal solutions for scenarios with a low number of periods, on larger scenarios the quality of the solutions quickly deteriorates and is far from optimality.

Table 2: Results of the B&C algorithm

Scenario	Upper bound	Lower bound	Gap (%)	Time (s)
20 / 6 / 1	3788.3	3788.3	0.0	2
20 / 6 / 2	4237.0	4237.0	0.0	1
20 / 6 / 3	4237.0	4237.0	0.0	302
20 / 12 / 1	8004.8	7261.7	9.3	10800
20 / 12 / 2	8826.9	8093.5	8.3	10800
20 / 12 / 3	7780.4	7572.0	2.7	10800
20 / 18 / 1	10844.0	8946.0	17.5	10800
20 / 18 / 2	11994.4	10385.0	13.4	10800
20 / 18 / 3	11415.6	9670.6	15.3	10800
40 / 6 / 1	6910.6	6910.6	0.0	13
40 / 6 / 2	8820.4	8820.4	0.0	7
40 / 6 / 3	7547.1	7374.2	2.3	10800
40 / 12 / 1	16706.5	10701.5	35.9	10800
40 / 12 / 2	15631.3	13407.7	14.2	10800
40 / 12 / 3	14391.1	10778.9	25.1	10800
40 / 18 / 1	31779.5	13170.6	58.6	10800
40 / 18 / 2	34152.3	16177.0	52.6	10800
40 / 18 / 3	41534.8	13053.2	68.6	10800
60 / 6 / 1	9982.3	9982.3	0.0	10266
60 / 6 / 2	11946.3	11946.3	0.0	115
60 / 6 / 3	10674.1	9645.9	9.6	10800
60 / 12 / 1	26298.4	14572.3	44.6	10800
60 / 12 / 2	22015.5	17407.6	20.9	10800
60 / 12 / 3	25747.5	14578.4	43.4	10800
60 / 18 / 1	62849.6	17784.2	71.7	10800
60 / 18 / 2	66632.2	21566.9	67.6	10800
60 / 18 / 3	59428.6	17478.4	70.6	10800
Average	20154.7	11094.4	24.2	8396.4

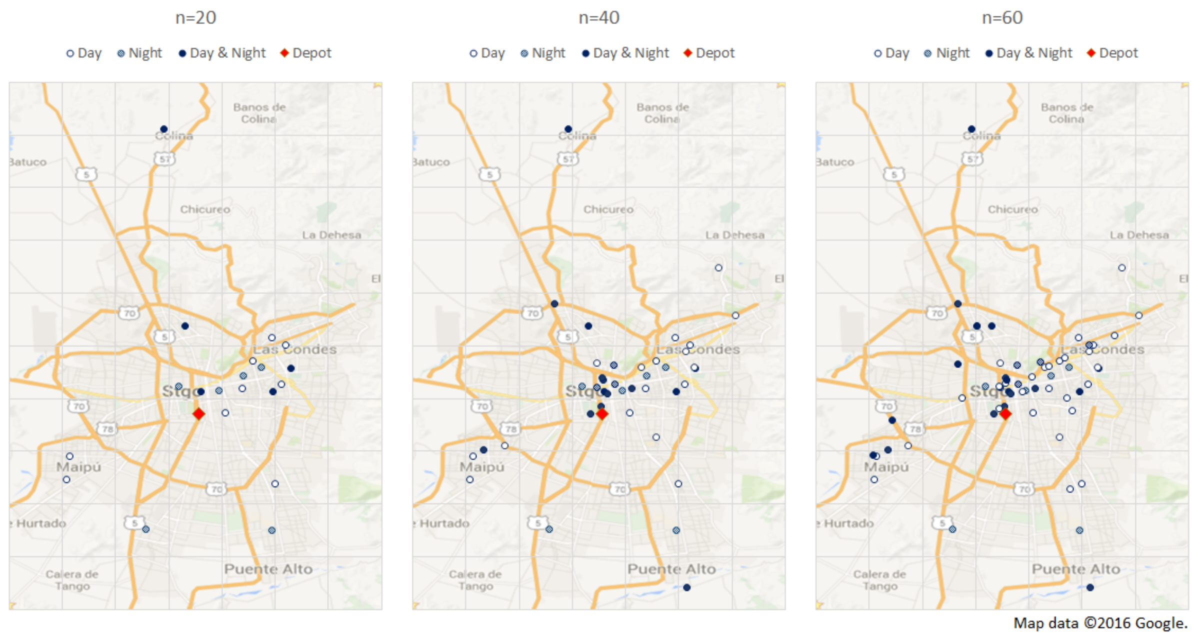


Figure 4: Locations of ATMs in Santiago for our scenarios

The results from the implementation of our VMNS matheuristic using $\alpha = 1$, i.e., giving both phases the same importance in terms of running time, are presented in Table 3. Execution time of the algorithm was also limited to three hours. These results show a general improvement in the upper bounds thanks to the local search structures, especially in bigger instances in which the upper bounds are reduced to less than half of their original value. In one particular scenario (20 / 12 / 2), though, the heuristic finds a slightly worse solution than the pure B&C. This means that the heuristic is best suited for bigger, more complex scenarios where the exact algorithm performs with greater difficulty.

An interesting result can be read from the new lower bounds shown in Table 3. Since our algorithm takes time away from B&C to perform local searches, and since the lower bounds for the overall problem can only be improved during the B&C stage, one could expect the new lower bounds to be worse than ones obtained from the pure B&C. However, in many scenarios the lower bounds actually improved. These are marked with an asterisk in Table 3. This can happen because tighter upper bounds can help prune the B&C tree, reducing the time employed exploring unfruitful nodes, thus speeding up the search for new nodes, sometimes leading to new lower bounds. The combination of improved solutions and sometimes tighter lower bounds has led to a significant improvement of the optimality gap, from an average of 24.2% to 14.7%.

As an example, we show the solution obtained using VMNS for scenario 40 / 18 / 1 in Figure 5. This figure shows the routes for the three vehicles in the nine out of 18 periods where routes are made. In this particular scenario, VMNS was able to improve the pure B&C upper bound solution by 38.48%.

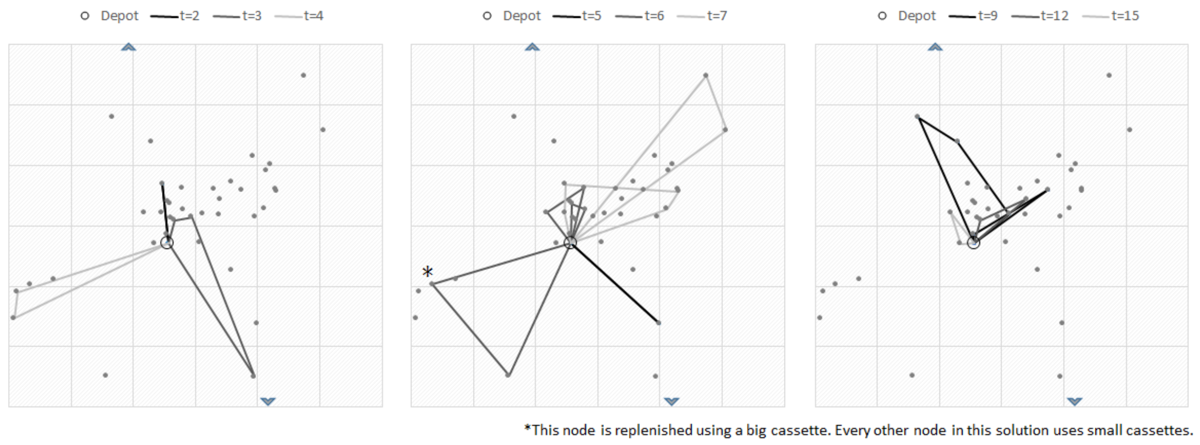

Depot
t=2
t=3
t=4
Depot
t=5
t=6
t=7
Depot
t=9
t=12
t=15
*
*This node is replenished using a big cassette. Every other node in this solution uses small cassettes.

Figure 5: Solution for scenario 40 / 18 / 1

In order to illustrate the efficiency of the VMNS, we show in Table 4 how much time it takes for VMNS to find a solution as good as the best one obtained by the exact algorithm. Note that the times shown here for B&C differ from the ones in Table 2 because here we report the first appearance of the best upper bound, not the total execution time. It can be seen here that our new algorithm works considerably faster requiring, on average, only 39.8% of the time consumed by B&C to find a solution at least as good, and in most cases, significantly better.

5.3 Sensitivity analysis

We have performed several experiments to test the behavior of our VMNS matheuristic for different values of parameter α, which defines how much time the algorithm invests performing local searches. In the limit where $\alpha \to 0$ the algorithm becomes a pure B&C. Table 5 shows the results obtained by the algorithm in medium-sized scenarios ($n = 40$)

Table 3: Results of the Variable MIP Neighborhood Search (VMNS) matheuristic

Scenario	Upper bound	Improvement over B&C (%)	Lower bound	Gap (%)	Time (s)
20 / 6 / 1	3788.3	0.0	3788.3	0.0	4
20 / 6 / 2	4237.0	0.0	4237.0	0.0	4
20 / 6 / 3	4237.0	0.0	4237.0	0.0	7
20 / 12 / 1	8004.8	0.0	7261.7	9.3	10800
20 / 12 / 2	8848.7	−0.2	8161.2*	8.5	10800
20 / 12 / 3	7771.7	0.1	7771.7*	2.6	10800
20 / 18 / 1	10829.2	0.1	9137.5*	17.4	10800
20 / 18 / 2	11995.3	0.0	10430.1*	13.4	10800
20 / 18 / 3	11377.7	0.3	9670.6	15.0	10800
40 / 6 / 1	6910.6	0.0	6910.6	0.0	63
40 / 6 / 2	8820.4	0.0	8820.4	0.0	22
40 / 6 / 3	7547.1	0.0	7547.1*	2.3	10800
40 / 12 / 1	13919.2	16.7	10701.6	23.1	10800
40 / 12 / 2	15437.2	1.2	13407.7	13.1	10800
40 / 12 / 3	13570.6	5.7	10941.9*	20.6	10800
40 / 18 / 1	19551.6	38.5	13170.6	32.6	10800
40 / 18 / 2	21007.1	38.5	16177.0	23.0	10800
40 / 18 / 3	20020.3	51.8	13053.2	34.8	10800
60 / 6 / 1	9982.3	0.0	9982.3	0.0	9439
60 / 6 / 2	11946.3	0.0	11946.3	0.0	361
60 / 6 / 3	10651.0	0.2	9646.0	9.4	10800
60 / 12 / 1	20318.3	22.7	14599.0*	28.3	10800
60 / 12 / 2	20507.7	6.9	17407.6	15.1	10800
60 / 12 / 3	18801.1	27.0	14578.4	22.5	10800
60 / 18 / 1	28296.8	55.0	17784.2	37.2	10800
60 / 18 / 2	30604.8	54.1	21566.9	29.5	10800
60 / 18 / 3	28840.0	51.5	17478.4	39.4	10800
Average	13993.4	13.70	10981.5	14.7	8366.7

* indicates an improvement with respect to the B&C lower bound.

Table 4: Time required by B&C to find its best solution, and time required by VMNS to find a solution at least as good

Scenario	B&C time (s)	VMNS time (s)	Fraction of time required (%)
20 / 6 / 1	2	2	100.0
20 / 6 / 2	1	1	100.0
20 / 6 / 3	1	2	200.0
20 / 12 / 1	8187	56	0.7
20 / 12 / 2	1700	0	0.0
20 / 12 / 3	689	120	17.4
20 / 18 / 1	7405	257	3.5
20 / 18 / 2	10373	0	0.0
20 / 18 / 3	10319	1279	12.4
40 / 6 / 1	1	3	300.0
40 / 6 / 2	8	10	125.0
40 / 6 / 3	4380	1	0.0
40 / 12 / 1	9512	83	0.9
40 / 12 / 2	8267	884	10.7
40 / 12 / 3	5959	193	3.2
40 / 18 / 1	9964	163	1.6
40 / 18 / 2	9901	109	1.1
40 / 18 / 3	6659	89	1.3
60 / 6 / 1	634	68	10.7
60 / 6 / 2	115	168	146.1
60 / 6 / 3	10086	2430	24.1
60 / 12 / 1	9399	353	3.8
60 / 12 / 2	9440	280	3.0
60 / 12 / 3	6343	235	3.7
60 / 18 / 1	9475	272	2.9
60 / 18 / 2	10007	88	0.9
60 / 18 / 3	9055	271	3.0
Average	5847.5	274.7	39.8

when α takes the value of 0.5, 1.0, and 2.0. We can observe that there is no apparent trend in the table regarding α, and that results are very similar among them, with an average deviation of only 1%. This means that the VMNS is very robust with respect to this parameter.

Table 5: Sensitivity analysis for parameter α

Scenario	UB $\alpha = 0.5$	UB $\alpha = 1.0$	UB $\alpha = 2.0$	Maximum deviation (%)
40 / 6 / 1	6910.6	6910.6	6910.6	0.0
40 / 6 / 2	8820.4	8820.4	8820.4	0.0
40 / 6 / 3	7547.1	7547.1	7547.1	0.0
40 / 12 / 1	13911.6	13919.2	13911.6	0.1
40 / 12 / 2	15358.1	15437.2	15488.6	0.8
40 / 12 / 3	13570.6	13570.6	13569.3	0.0
40 / 18 / 1	18914.5	19551.6	19197.9	3.4
40 / 18 / 2	21063.2	21007.1	21821.7	3.9
40 / 18 / 3	19889.8	20020.3	19969.9	0.7
Average	13998.4	14087.1	14137.5	1.0

To further show this point, in Figure 6 we present the evolution of the solution over time for different versions of the algorithm, for scenarios with $H = 12$. This figure shows two interesting things. First, looking at how similar the evolution of the solution is for the different values of α, it confirms that the algorithm is not very sensitive to this parameter. This means that it is not necessary to spend too much effort calibrating this value on future implementations of this algorithm. Second, irrespective of the value of α, the figure also shows that the VMNS algorithm helps reaching very good solutions in significantly less time than B&C, and that it works specially well during the first minutes of execution. Extending the maximum running time yields only very marginal improvements.

6 Conclusions

In this paper we have solved a practical problem arising in the ATM management and replenishments in Santiago, Chile. This rich and challenging problem shares much of its structure with the IRP, but some features makes it unique. We have modeled the problem and solved it by branch-and-cut. In order to efficiently tackle large instances, we have derived and formalized a Variable MIP Neighborhood Search heuristic, which takes advantage of the MIP implementation. Our VMNS is shown to significantly improve the solutions using only a fraction of the time. As a positive side effect, our heuristic has also improved several lower bounds, helping closing the gap.

The VMNS has several features that make it appropriate to solve other classes of problems. First, it relies on the mathematical formulation of the problem, and helps solving it when the instance size is excessive for a branch-and-bound (or branch-and-cut) method to tackle it. Second, deriving neighborhoods is as simple as imposing a few new constraints on the model, which are to be designed as problem-specific, even though our neighborhoods can be used as general frameworks for many distribution problems. Third, based on a single parameter, one can drive the VMNS to act more like a pure local search heuristic or like a pure exact method. Finally, given enough time, the VMNS is guaranteed to terminate and is capable of proving optimality (relying on the underlying branch-and-bound or -cut method), thus, being an exact algorithm.

References

[1] T. F. Abdelmaguid and M. M. Dessouky. A genetic algorithm approach to the integrated inventory-distribution problem. *International Journal of Production Research*, 44(21):4445–4464, 2006.

[2] E.-H. Aghezzaf, B. Raa, and H. van Landeghem. Modeling inventory routing problems in supply chains of high consumption products. *European Journal of Operational Research*, 169(3):1048–1063, 2006.

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

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

[5] C. Archetti, L. Bertazzi, G. Paletta, and M. G. Speranza. Analysis of the maximum level policy in a production-distribution system. *Computers & Operations Research*, 12(38):1731–1746, 2011.

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

[7] T. Bektaş, E. Demir, and G. Laporte. Green vehicle routing. In H. N. Psaraftis, editor, *Green Transportation Logistics*, volume 226 of *International Series in Operations Research & Management Science*, pages 243–265. Switzerland, 2016.

[8] L. Bertazzi, G. Paletta, and M. G. Speranza. Deterministic order-up-to level policies in an inventory routing problem. *Transportation Science*, 36(1):119–132, 2002.

[9] L. Bertazzi, A. Bosco, F. Guerriero, and D. Laganà. A stochastic inventory routing problem with stock-out. *Transportation Research Part C: Emerging Technologies*, 27 (1):89–107, 2013.

[10] C. Blum, P. Pinacho, M. López-Ibáñez, and J. A. Lozano. Construct, merge, solve & adapt a new general algorithm for combinatorial optimization. *Computers & Operations Research*, 68:75–88, 2016.

[11] L. C. Coelho and G. Laporte. The exact solution of several classes of inventory-routing problems. *Computers & Operations Research*, 40(2):558–565, 2013.

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

[13] L. C. Coelho, J.-F. Cordeau, and G. Laporte. The inventory-routing problem with transshipment. *Computers & Operations Research*, 39(11):2537–2548, 2012.

[14] L. C. Coelho, J.-F. Cordeau, and G. Laporte. Thirty years of inventory-routing. *Transportation Science*, 48(1):1–19, 2014.

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

[16] G. Desaulniers, J. G. Rakke, and L. C. Coelho. A branch-price-and-cut algorithm for the inventory-routing problem. *Transportation Science*, forthcoming, 2015.

[17] V. Maniezzo, T. Stützle, and S. Voß. *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*. Springer, New York, 2009.

[18] S. Mirzaei and A. Seifi. Considering lost sale in inventory routing problems for perishable goods. *Computers & Industrial Engineering*, forthcoming, 2015.

[19] A. Montoya, C. Guéret, J. E. Mendoza, and J. G. Villegas. A multi-space sampling heuristic for the green vehicle routing problem. *Transportation Research Part C: Emerging Technologies*, forthcoming, 2015.

[20] E.-G. Talbi. Combining metaheuristics with mathematical programming, constraint programming and machine learning. *4OR*, 11(2):101–150, 2013.

[21] R. G. van Anholt, L. C. Coelho, G. Laporte, and I. F. A. Vis. An inventory-routing problem with pickups and deliveries arising in the replenishment of automated teller machines. *Transportation Science*, forthcoming, 2015.

[22] M. Vidović, D. Popović, and B. Ratković. Mixed integer and heuristics model for the inventory routing problem in fuel delivery. *International Journal of Production Economics*, 147(Part C):593–604, 2014.

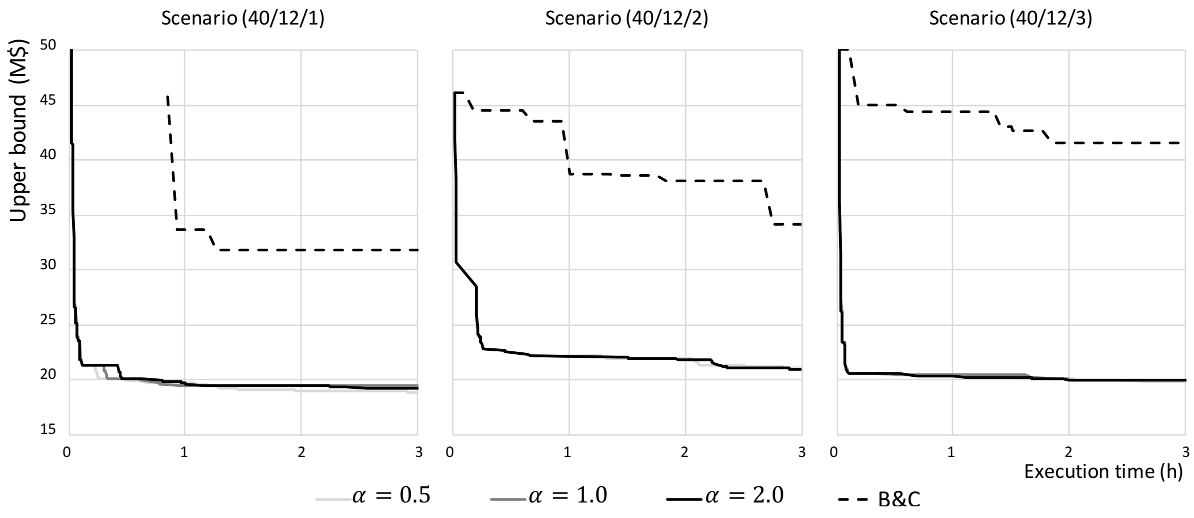


Figure 6: Convergence of the algorithms