CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

An Inventory-Routing Problem with Pickups and Deliveries Arising in the Replenishment of Automated Teller Machines

Roel G. van Anholt
Leandro C. Coelho
Gilbert Laporte
Iris F.A. Vis

November 2013

CIRRELT-2013-71

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

An Inventory-Routing Problem with Pickups and Deliveries Arising in the Replenishment of Automated Teller Machines

Roel G. van Anholt[1], Leandro C. Coelho[2,3,*], Gilbert Laporte[2,4], Iris F.A. Vis[5]

[1] VU University Amsterdam, Faculty of Economics and Business Administration, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[3] Department of Operations and Decision Systems, Université Laval, 2325, de la Terrasse, Québec, Canada G1V 0A6

[4] Department of Management Sciences, HEC Montréal, 3000 Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

[5] University of Groningen, Faculty of Economics and Business, Nettelbosje 2, 9747 AE Groningen, The Netherlands

Abstract. The purpose of this paper is to introduce, model and solve a rich multi-period inventory-routing problem with simultaneous pickups and deliveries. Commodities can be brought from and to the depot, as well as being exchanged among customers to efficiently manage their inventory shortages and surpluses. A single customer can both provide and receive commodities at different periods, since its demand changes dynamically throughout the planning horizon and can be either positive or negative. This problem arises, for instance, in the replenishment operations of automated teller machines. New technology provides these machines with the additional functionality of receiving deposits and reissuing these to subsequent customers. Motivated by a real case in the Netherlands, we formulate the problem as a mixed-integer linear programming model and propose an exact branch-and-cut algorithm for its resolution. Given the complexity of the problem, we also propose a flexible clustering heuristic. Through extensive computational experiments using real data, we assess the performance of the solution algorithm and of the clustering procedure. The results show that we are able to obtain good lower and upper bounds for this new and challenging practical problem.

Keywords. Inventory-routing, vehicle routing, inventory management, pickup and delivery, branch-and-cut, clustering, exact algorithm, recirculation automated teller machines.

Acknowledgements. This work was partly supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant 39682-10. This support is gratefully acknowledged. We also thank Calcul Québec for providing parallel computing facilities.

---

* Corresponding author: Leandro.Coelho@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2013

1 Introduction

The purpose of this paper is to introduce, model and solve an inventory-routing problem with pickups and deliveries (IRPPD) arising in the replenishment of automated teller machines (ATMs). Our study is motivated by the problem faced by a transporter responsible for such operations in the Netherlands. As in other cash-intensive economies, the Dutch credit institutions are gradually replacing regular ATMs by recirculation ATMs (RATMs), which are capable of accepting and dispensing banknotes, as well as checking their quality and authenticity. RATMs therefore provide customers the capability of both depositing and withdrawing cash. These machines provide tangible benefits to banks and customers, and are becoming the new standard in many markets. Compared with the situation that prevailed two years earlier, the number of installations worldwide has globally increased by 45%, to reach 670,000 in 2011, and this number is expected to double by 2017 [46]. RATMs allow retailers to make deposits and enable customers to later withdraw the same cash. In this sense, RATMs are considerably more self-sufficient than the regular machines which can only dispense cash. RATMs only require a visit to prevent the inventory level from reaching zero or from attaining the holding capacity of the machines when the supply and demand are out of sync. When the RATM is empty, a customer can only use it to deposit cash, and when it is full only withdraws are allowed.

In the problem considered here, the RATMs are replenished or partly emptied by using a fleet of rented armoured trucks based at a single depot. These trucks deliver cash from the depot to some machines, collect cash from some others to bring it back to the depot, or transfer cash between machines. The last operation reduces the routing cost and sometimes allows smaller or fewer trucks to be used. An important feature of the problem is the presence of inventory holding costs. Indeed, cash lying in a machine generates an implicit holding cost partly because it is insured and also because it incurs lost interest income.

To keep RATMs fully operational, that is when customers can use the RATM to both

deposit and withdraw cash, one must solve an inventory-routing problem with pickups and deliveries (IRPPD). The IRPPD combines the features of two well-known classes of the vehicle routing problem: the inventory-routing problem (IRP) and the pickup and delivery problem (PDP) which we now briefly review.

The IRP belongs to the broader field of vendor-managed inventory systems in which a supplier coordinates the inventory management of a number of locations [20]. In IRPs, the supplier must simultaneously decide when to visit its inventory locations, how much to deliver to each of them, and how to combine the deliveries into vehicle routes. There exists numerous variants of this problem [1, 20] and the related literature is quickly expanding. The first exact algorithm for the single-vehicle IRP was based on branch-and-cut [5]. Archetti et al. [6] later presented a hybrid tabu search matheuristic algorithm capable of dealing with larger instances, and yielding solutions with very low optimality gaps. Coelho et al. [19] presented an adaptive large neighbourhood search heuristic for the multi-vehicle IRP, and Coelho and Laporte [16] were the first to solve the multi-vehicle IRP exactly. Recently, multi-vehicle and multi-commodity IRPs were also solved to optimality by Coelho and Laporte [17]. All these algorithms are based on branch-and-cut. For a recent survey of models and algorithms, see Coelho et al. [20].

A related problem appears in maritime transportation, in which ships have to visit several ports to continuously deliver and pickup merchandise and commodities. Applications include the distribution of cement [14], chemical products [24], liquefied gases [44], among others. For reviews of maritime transportation, see Christiansen et al. [13] and Christiansen et al. [15].

The PDP concerns the collection and distribution of one or several commodities from and to a set of locations. Berbeglia et al. [10] classify PDPs and distinguish between three different problem structures: *many-to-many* (M-M), *one-to-many-to-one* (1-M-1) and *one-to-one* (1-1). The M-M structure means that each commodity may have multiple origins and multiple destinations, and that each location may be the origin or destination of multiple commodities. In 1-M-1 problems, some commodities are picked up at the

depot and transported to some locations, while other commodities are picked up at these locations and transported to the depot. The 1-1 structure refers to a context in which each commodity has a single origin and a single destination, like in dial-a-ride problems [22].

Two important classes of PDPs with an M-M structure are the Swapping Problem (SP) and the 1-commodity Pickup and Delivery Traveling Salesman Problem (1-PDTSP). In the SP, introduced by Anily and Hassin [3], each vertex of a graph provides a commodity and requests a commodity, possibly the same one. The problem is to design a least cost vehicle route in order to satisfy all requests. This problem is NP-hard on general graphs, but polynomial on some special structures [4]. Erdoğan et al. [28] have developed heuristics and a branch-and-cut algorithm for the multi-vehicle case. In the 1-PDTSP, each vertex either provides or requests a given amount of a single commodity, and a single vehicle route must be designed in order to transfer the right amounts of the commodity among vertices. The problem was introduced by Hernández-Pérez and Salazar-González [35] and was solved by branch-and-cut [35, 36, 38] and by heuristics [37, 51]. The 1-PDTSP arises in the rebalancing operations in shared bicycle systems [9, 12, 21, 30, 29, 45]. This application is similar to the problem encountered in the replenishment of RATMs in the sense that in both cases the aim is to shuffle some commodities between locations so as to bring their inventory level within a given interval (see Erdoğan et al. [29]). Some algorithms [21, 45] are capable of handling the multi-vehicle case.

The PDP with a 1-M-1 structure deals with two types of commodities. Delivery commodities are transported from a single depot to multiple nodes, and pickup commodities are transported from multiple nodes back to the depot. A typical application is the recycling of products such as beer bottles, pallets and containers [10]. A distinction is made between combined and single demands. Combined demands occur when locations may require both a pickup and a delivery, whereas single demand refers to problems where each location has either a pickup or delivery demand. The single demand problem was introduced by Mosheiov [42]. Heuristics were proposed by Gendreau et al. [32] and by

Subramanian and Battarra [47], whereas Baldacci et al. [7] and Hernández-Pérez and Salazar-González [38] solved the problem exactly by branch-and-cut. The combined demand case was introduced by Min [41]. It was solved heuristically by e.g., Bianchessi and Righini [11], Subramanian et al. [48], Zachariadis et al. [50], Vidal et al. [49], and exactly by branch-and-price algorithms by Angelelli and Mansini [2], Dell'Amico et al. [25], Subramanian and Battarra [47]. We are not aware of any contributions on the multi-commodity 1-M-1 PDP.

Our aim is to model and solve the IRPPD arising in the replenishment of RATMs by means of an exact branch-and-cut algorithm. The paper makes four main contributions. First, it introduces pickups and deliveries within an IRP context. Second, it combines two PDP structures: the 1-M-1 structure which accounts for commodity movements from the depot to RATMs to the depot, and the M-M structure which refers to commodity transfers among RATMs. Hence, our PDP could appropriately be designated as a 1-M-M-1 problem. Note that we tackle a rather general case for realistic sized problems, in that there are several vehicles and side constraints in addition to the standard IRP and PDP features. Third, to cope with realistic sized problems, we propose a clustering heuristic which can be applied prior to executing the algorithm. Our fourth contribution is to apply our algorithm to a real-world case arising in the Netherlands.

The remainder of this paper is organized as follows. We formally describe the problem in mathematical terms in Section 2, where we provide a mixed-integer formulation. The branch-and-cut algorithm is described in Section 3. The clustering heuristic is detailed in Section 4, followed by the results of extensive computational results in Section 5, and by conclusions in Section 6.

2 Mathematical Programming Formulation

The IRPPD is defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V} = \{0, ..., n\}$ is the vertex set and $\mathcal{A} = \{(i, j) : i, j \in \mathcal{V}, i \neq j\}$ is the arc set. Vertex 0 represents the depot

and the vertices of $\mathcal{V}' = \mathcal{V} \setminus \{0\}$ represent RATM locations. Each RATM incurs unit inventory holding costs α_i per period ($i \in \mathcal{V}'$), and has an inventory holding capacity C_i. A handling cost β is incurred at the depot, and is proportional to the quantity picked up and delivered at the depot. The length of the planning horizon is p, with discrete time periods $t \in \mathcal{T} = \{1, \ldots, p\}$. A set of rented armoured vehicles $k \in \mathcal{K} = \{1, \ldots, K\}$ is available, each with capacity Q_k and average speed s_k. A renting cost γ_k per period is incurred if vehicle k is used. Each vehicle is able to perform one route per period, from the depot to a subset of RATMs, each requiring r units of time to be served, and back to the depot. The shift of each vehicle is limited to S time units, after which δ monetary units per unit of overtime are incurred. A routing cost c_{ij} is associated with arc $(i, j) \in \mathcal{A}$. We assume the depot has sufficient inventory and capacity to perform all pickups and deliveries during the planning horizon. The inventories are not allowed to exceed the holding capacity nor are they allowed to become negative. At the beginning of the planning horizon the decision maker knows the current inventory level I_i^0 of the RATMs, and receives information on the net demand d_i^t of each RATM i for each period t. Negative demands mean that the RATM provides a commodity, while positive demands mean that the RATM receives some quantity of the commodity. We assume that the quantities q_i^t received by RATM i in period t can be used to satisfy its net demand in that period. The quantities picked up at the depot and at the RATMs may be delivered to any RATM to satisfy their demands. The objective of the problem is to minimize the total cost while satisfying the net demand for each RATM in each period.

The variables used in the formulation are as follows. Four families of binary variables are used: directed routing variables x_{ij}^{kt} are equal to 1 if and only if arc (i, j) is used on the route of vehicle k in period t; visiting variables y_i^{kt} are equal to 1 if and only if RATM i is visited by vehicle k in period t; delivery variables w_i^{kt} are equal to 1 if and only if a delivery is made to RATM i by vehicle k in period t; and pickup variables z_i^{kt} are 1 if and only if a pickup is performed at RATM i by vehicle k in period t. Integer variables I_i^t represent the inventory level at RATM $i \in \mathcal{V}'$ at the end of period $t \in \mathcal{T}$. Variables q_i^{kt} are

integers representing the product quantity delivered to RATM i using vehicle k in period t, and variables p_i^{kt} are integers representing the product quantity picked up from RATM i using vehicle k in period t. Two sets of variables represent the amount of inventory carried by vehicle k in period t out of and into the depot: J_k^t and H_k^t, respectively. The load of vehicle k after serving RATM i in period t is represented by variables u_i^{kt}, and E_k^t represents the overtime in number of extra minutes worked by vehicle k in period t over the maximum shift duration S.

The problem is then formulated as follows:

$$\text{minimize} \sum_{i\in\mathcal{V}'}\sum_{t\in\mathcal{T}} \alpha_i I_i^t + \sum_{k\in\mathcal{K}}\sum_{t\in\mathcal{T}} \beta\left(J_k^t + H_k^t\right) + \sum_{k\in\mathcal{K}}\sum_{t\in\mathcal{T}} \gamma_k y_0^{kt} + \sum_{k\in\mathcal{K}}\sum_{t\in\mathcal{T}} \delta E_k^t, \tag{1}$$

subject to the following constraints:

$$I_i^t = I_i^{t-1} + \sum_{k\in\mathcal{K}} q_i^{kt} - \sum_{k\in\mathcal{K}} p_i^{kt} + d_i^t \quad i \in \mathcal{V}' \quad t \in \mathcal{T} \tag{2}$$

$$0 \le I_i^t \le C_i \quad i \in \mathcal{V}' \quad t \in \mathcal{T} \tag{3}$$

$$\sum_{j\in\mathcal{V}, i<j} x_{ij}^{kt} + \sum_{j\in\mathcal{V}, j<i} x_{ji}^{kt} = 2y_i^{kt} \quad i \in \mathcal{V} \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{4}$$

$$\sum_{i\in\mathscr{S}}\sum_{j\in\mathscr{S}} x_{ij}^{kt} \le \sum_{i\in\mathscr{S}} y_i^{kt} - y_m^{kt} \quad \mathscr{S} \subseteq \mathcal{V} \quad k \in \mathcal{K} \quad t \in \mathcal{T} \quad m \in \mathscr{S} \tag{5}$$

$$w_i^{kt} \le y_i^{kt} \quad i \in \mathcal{V}' \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{6}$$

$$z_i^{kt} \le y_i^{kt} \quad i \in \mathcal{V}' \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{7}$$

$$q_i^{kt} \le w_i^{kt}\left(C_i - I_i^t\right) \quad i \in \mathcal{V}' \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{8}$$

$$p_i^{kt} \le z_i^{kt} I_i^t \quad i \in \mathcal{V}' \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{9}$$

$$w_i^{kt} + z_i^{kt} \le 1 \quad i \in \mathcal{V}' \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{10}$$

$$s_k \sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij}^{kt} + r\sum_{i\in\mathcal{V}'} y_i^{kt} \le S + E_k^t \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{11}$$

$$u_j^{kt} \ge \left(u_i^{kt} + p_j^{kt} - q_j^{kt}\right) x_{ij}^{kt} \quad i \in \mathcal{V} \quad j \in \mathcal{V} \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{12}$$

$$0 \leq u_i^{kt} \leq Q_k \quad i \in \mathcal{V} \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{13}$$

$$J_k^t = u_0^{kt} y_0^{kt} \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{14}$$

$$H_k^t = \sum_{i \in \mathcal{V}'} x_{i0}^{kt} u_i^{kt} \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{15}$$

$$q_i^{kt}, p_i^{kt}, J_k^t, H_k^t \geq 0 \quad i \in \mathcal{V}' \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{16}$$

$$x_{ij}^{kt} \in \{0,1\} \quad (i,j) \in \mathcal{A} \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{17}$$

$$y_i^{kt} \in \{0,1\} \quad i \in \mathcal{V} \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{18}$$

$$w_i^{kt}, z_i^{kt} \in \{0,1\} \quad i \in \mathcal{V}' \quad k \in \mathcal{K} \quad t \in \mathcal{T}. \tag{19}$$

The objective function (1) minimizes the total cost of inventory holding, inventory handling at the depot, and vehicle renting. Constraints (2) state the inventory conservation condition over successive periods: they define the inventory in period t as the inventory held in period $t-1$, plus the quantity delivered, minus the quantity picked up, plus the net demand of the location. Constraints (3) define the bounds on the inventory held by each RATM throughout all periods. Constraints (4) and (5) guarantee that proper vehicle routes are created: they are degree and subtour elimination constraints, respectively. Constraints (6) link delivery binary variables w_i^{kt} to visiting binary variables y_i^{kt}. They allow a delivery decision to be made only if a visit is performed to the RATM. Constraints (7) are similar to (6) and apply to the pickup decisions. Constraints (8) and (9) allow a quantity to be delivered or picked up at an RATM only if the corresponding binary decision is set to one. Constraints (10) ensure that a visit to an RATM is used to either a pickup or a delivery operation. Constraints (11) guarantee that the shift duration is respected and that overtime is properly accounted. Constraints (12) ensure that the load within the vehicle is consistent along its route. Constraints (13) set bounds on the load inside the vehicle after serving RATM i. Constraints (14) set the load of the vehicle when leaving the depot, while constraints (15) set its load when returning to the depot at the end of the route. Finally, constraints (16)−(19) define non-negativity and binary conditions on the variables.

Three comments are relevant with respect to this formulation. The first regards constraints (12), which resemble the Miller-Tucker-Zemlin subtour elimination constraints [40]. These constraints do not eliminate subtours in this context, because the load within the vehicle is not monotonically increasing or decreasing. We therefore impose Dantzig-Fulkerson-Johnson subtour elimination constraints (5) whose number is exponential in n [23]. Second, the load consistency constraints (12) are used in other pickup and delivery problems, see, e.g., Desaulniers et al. [26], Gribkovskaia et al. [34], Hoff et al. [39]. When it is known in advance whether an RATM requires a pickup or a delivery, they can been lifted [34]. However, this is not the case here. The third comment refers to the fact that the formulation is non-linear due to constraints (8), (9), (12), (14) and (15). However, these can be linearized as follows. Constraints (8) and (9) can be rewritten in a slightly weaker form as

$$q_i^{kt} \leq w_i^{kt} C_i \quad i \in \mathcal{V}' \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{20}$$

$$p_i^{kt} \leq z_i^{kt} C_i \quad i \in \mathcal{V}' \quad k \in \mathcal{K} \quad t \in \mathcal{T}. \tag{21}$$

On their own, these constraints would allow quantities to be picked up or delivered to violate the inventory bounds, but together with constraints (3), they are feasible linear representations of constraints (8) and (9). Constraints (12) can be linearized as

$$u_j^{kt} \geq u_i^{kt} + p_j^{kt} - q_j^{kt} - \left(1 - x_{ij}^{kt}\right) Q_k \quad i \in \mathcal{V} \quad j \in \mathcal{V} \quad k \in \mathcal{K} \quad t \in \mathcal{T}. \tag{22}$$

Finally, constraints (14) and (15) can be rewritten in a linear form as

$$J_k^t = u_0^{kt} \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{23}$$

$$H_k^t \geq u_i^{kt} - \left(1 - x_{i0}^{kt}\right) Q_k \quad i \in \mathcal{V}' \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{24}$$

$$H_k^t \leq Q_k \quad k \in \mathcal{K} \quad t \in \mathcal{T}. \tag{25}$$

This formulation can be strengthened by imposing the following valid inequalities:

$$x_{ij}^{kt} \leq y_i^{kt} \quad i,j \in \mathcal{V}' \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{26}$$

$$y_i^{kt} \leq y_0^{kt} \quad i \in \mathcal{V}' \quad k \in \mathcal{K} \quad t \in \mathcal{T}. \tag{27}$$

Constraints (26) are referred to as logical inequalities. They tighten the relation between routing and visiting variables. Constraints (27) include the supplier in the route of vehicle k if any RATM is visited by that vehicle in that period.

3 Branch-and-Cut Algorithm

We have implemented a branch-and-cut algorithm capable of solving the formulation we have introduced. All variables of the formulation are explicitly handled by the algorithm, but we cannot generate all subtour elimination constraints (5) a priori. These will be dynamically generated as cuts as they are found to be violated. The formulation is then solved by branch-and-cut as follows. At a generic node of the search tree, a linear program with relaxed integrality constraints is solved, a search for violated constraints is performed, and violated valid inequalities are added to the current program which is then reoptimized. This process is reiterated until a feasible or dominated solution has been reached, or until no more cuts can be added. At this point, branching on a fractional variable occurs. We provide in Algorithm 1 a sketch of the branch-and-cut scheme. Details regarding the implementation and improvements to this algorithm are provided in Section 5.2.

4 Clustering Heuristic

In order to solve realistic sized problems we design a clustering heuristic which is used prior to executing the branch-and-cut algorithm. This heuristic decides for each RATM $i \in \mathcal{V}'$ and each period $t \in \mathcal{T}$ whether it must be visited, must not be visited, or must perhaps be visited. Specifically, for each period t we distinguish five subsets, a *must*

Algorithm 1 Branch-and-cut algorithm

At the root node of the search tree, generate and insert all valid inequalities into the program.
$z^* \leftarrow \infty$.
Termination check:
if there are no more nodes to evaluate **then**
  Stop with the incumbent and optimal solution of cost z^*.
else
  Select one node from the branch-and-bound tree.
end if
Subproblem solution: solve the LP relaxation of the node and let z be its cost.
if the current solution is feasible **then**
  if $z \geq z^*$ **then**
    Go to termination check.
  else
    $z^* \leftarrow z$.
    Update the incumbent solution.
    Prune the nodes with a lower bound larger than or equal to z^*.
    Go to termination check.
  end if
end if
Cut generation:
if the solution of the current LP relaxation violates any cuts **then**
  Identify connected components as in Padberg and Rinaldi [43].
  Determine whether the component containing the supplier is weakly connected as in Gendreau et al. [31].
  Add violated subtour elimination constraints (5).
  Go to subproblem solution.
end if
Branching: branch on one of the fractional variables.
Go to the termination check.

pickup set $\mathcal{U}^t$, a *must deliver* set $\mathcal{W}^t$, a *perhaps pickup* set $\mathcal{X}^t$, a *perhaps deliver* set $\mathcal{Y}^t$, and a *not visit* set $\mathcal{Z}^t$. These five sets define a partition of $\mathcal{V}'$.

The aim of the clustering heuristic is to (greatly) reduce the branch-and-cut algorithm's calculation time by fixing several binary variables prior to its execution. When RATMs are added to one of the subsets $\mathcal{U}^t$, $\mathcal{W}^t$, or $\mathcal{Z}^t$, the size of the formulation is greatly reduced and the branch-and-cut algorithm benefits from this reduction, because some of the decisions related to pickups, deliveries, and visits for some RATMs are already made. When RATMs are added to one of the other two subsets $\mathcal{X}^t$ or $\mathcal{Y}^t$, the algorithm still decides for each period t whether a visit is required, but the benefit of the clustering heuristic is that it limits the choice to either a pickup or a delivery. These steps are implemented as follows:

$$\sum_{k\in\mathcal{K}} z_i^{kt} = 1 \quad i \in \mathcal{U}^t \quad t \in \mathcal{T} \tag{28}$$

$$\sum_{k\in\mathcal{K}} w_i^{kt} = 1 \quad i \in \mathcal{W}^t \quad t \in \mathcal{T} \tag{29}$$

$$\sum_{k\in\mathcal{K}} w_i^{kt} = 0 \quad i \in \mathcal{X}^t \quad t \in \mathcal{T} \tag{30}$$

$$\sum_{k\in\mathcal{K}} z_i^{kt} = 0 \quad i \in \mathcal{Y}^t \quad t \in \mathcal{T} \tag{31}$$

$$\sum_{k\in\mathcal{K}} y_i^{kt} = 0 \quad i \in \mathcal{Z}^t \quad t \in \mathcal{T}. \tag{32}$$

Also, it follows directly that several other constraints fixing binary variables can be derived from (28)–(32). These are used to further reduce the size of the problem by effectively eliminating several variables from the problem:

$$\sum_{k\in\mathcal{K}} y_i^{kt} = 1 \quad i \in \mathcal{U}^t \quad t \in \mathcal{T} \tag{33}$$

$$w_i^{kt} = 0 \quad i \in \mathcal{U}^t \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{34}$$

$$\sum_{k\in\mathcal{K}} y_i^{kt} = 1 \quad i \in \mathcal{W}^t \quad t \in \mathcal{T} \tag{35}$$

$$z_i^{kt} = 0 \quad i \in \mathcal{W}^t \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{36}$$

$$w_i^{kt} = 0 \quad i \in \mathcal{X}^t \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{37}$$

$$z_i^{kt} = 0 \quad i \in \mathcal{Y}^t \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{38}$$

$$w_i^{kt} = 0 \quad i \in \mathcal{Z}^t \quad k \in \mathcal{K} \quad t \in \mathcal{T} \tag{39}$$

$$z_i^{kt} = 0 \quad i \in \mathcal{Z}^t \quad k \in \mathcal{K} \quad t \in \mathcal{T}. \tag{40}$$

The pseudo-code for our clustering heuristic is presented in Algorithm 2. Four parameters are defined below, which allow us to test various clustering settings. The first parameter relates to *due-periods* and *tipping-periods*. A due-period is a period in which a visit is required to prevent the inventory from exceeding the holding capacity or from a stock-out. A tipping-period is a period in which a pickup is required to remain cost efficient, that is when the holding cost is disproportionately high related to the cost of visiting the RATM.

- m: number of periods preceding a due-period or a tipping-period
- f: number of RATMs which are closest to RATM i
- g: minimum inventory level of RATM i, in a percentage of its holding capacity C_i
- b: expected number of RATM visits per vehicle k in a period t

In what follows, let $\bar{I}_i^t = I_i^0 + \sum_{t=1}^{t'} d_i^t$ be the cumulative inventory of RATM i in period t'. This is useful to identify up to which period t' the RATM will respect its inventory constraints without intervention from the depot.

The clustering heuristic starts by adding all RATMs i in all periods t to the not visit subset $\mathcal{Z}^t$ (see lines 2 – 4 in Algorithm 2). When period $t = 1$ or when parameter $m = 0$, lines 5 – 10 ensure that RATMs are added to the must visit subsets $\mathcal{U}^t$ and $\mathcal{W}^t$ when the holding capacity C_i is exceeded or when the inventory $\bar{I}_i^{t'}$ becomes negative. If either condition is satisfied, the algorithm does not have the flexibility to choose between preceding or succeeding periods to visit the RATM in. Therefore, the RATM must be visited.

Algorithm 2 Clustering heuristic

$\bar{\mathcal{V}}' \leftarrow \mathcal{V}'$.
for all i **do**
  for all t **do**
    Add i to $\mathcal{Z}^t$.
    if $t = 1$ **or** $m = 0$ **then**
      if $\bar{I}_i^{t'} > C_i$ **then**
        Add i to $\mathcal{U}^t$; remove i from $\mathcal{Z}^t$; remove i from $\bar{\mathcal{V}}'$.
      else if $\bar{I}_i^{t'} < 0$ **then**
        Add i to $\mathcal{W}^t$; remove i from $\mathcal{Z}^t$; remove i from $\bar{\mathcal{V}}'$.
      end if
    else
      for $t' = t - m$ to $t' = t; t' > 1$ **do**
        if $(\sum_{t \in \mathcal{T}} d_i^t)/t > 0$ **and** $(\bar{I}_i^{t'} \times \alpha_i) > (\gamma_k/b/(\bar{I}_i^{t'} < 0/((\sum_{t \in \mathcal{T}} d_i^t)/t)))$ **then**
          Add i to $\mathcal{X}^t$; remove i from $\mathcal{Z}^t$; remove i from $\bar{\mathcal{V}}'$.
        else if $\bar{I}_i^{t'} > C_i$ **then**
          Add i to $\mathcal{X}^t$; remove i from $\mathcal{Z}^t$; remove i from $\bar{\mathcal{V}}'$.
        else if $\bar{I}_i^{t'} < 0$ **then**
          Add i to $\mathcal{Y}^t$; remove i from $\mathcal{Z}^t$; remove i from $\bar{\mathcal{V}}'$.
        end if
      end for
    end if
    for all f number of RATMs j which have the smallest c_{ij} to i **do**
      if $i \in \{\mathcal{U}^t, \mathcal{X}^t\}$ **and** $(\bar{I}_j^{t'}/C_j) < (1 - g)$ **then**
        Add j to $\mathcal{Y}^t$; remove i from $\mathcal{Z}^t$; remove i from $\bar{\mathcal{V}}'$.
      else if $i \in \{\mathcal{W}^t, \mathcal{Y}^t\}$ **and** $(\bar{I}_j^{t'}/C_j) > g$ **then**
        Add j to $\mathcal{X}^t$; remove i from $\mathcal{Z}^t$; remove i from $\bar{\mathcal{V}}'$.
      end if
    end for
  end for
end for
Return $\mathcal{U}^t, \mathcal{W}^t, \mathcal{X}^t, \mathcal{Y}^t, \mathcal{Z}^t$.

If the condition in line 5 is not met, we check three more conditions in lines 13 – 18 in order to add RATMs i to the perhaps visit subsets $\mathcal{X}^t$ and $\mathcal{Y}^t$ in period t as well as to the subsets in m number of preceding periods t. Line 12 ensures that RATMs i are also added to the subsets in m number of preceding periods t. We introduce lines 13 and 14 to ensure RATM i is visited for a pickup when a tipping-period occurs, i.e., when the holding cost exceeds the cost of visiting the RATM. More precisely, the following condition is verified: the inventory $\bar{I}_i^{t'}$ of RATM i in period t multiplied with the unit inventory holding cost α_i exceeds the vehicle rental cost γ_k divided by the expected number of visits per route b and divided by the expected number of days elapsed since the last visit. To this end, we calculate the expected number of days elapsed since the last visit by dividing the inventory level $\bar{I}_i^t$ by the average demand $\sum_{t \in \mathcal{T}} d_i^t/t$. If the latter condition is met, then the RATM i is added to the perhaps pickup set $\mathcal{X}^t$. RATMs i are also added to the $\mathcal{X}^t$ subset in periods t when the inventory exceeds capacity $\bar{I}_i^{t'} > C_i$ (see lines 15 – 16). In the case of stockouts, the RATMs are added to the perhaps deliver subset $\mathcal{Y}^t$, which is indicated in lines 17 and 18.

Most likely, some RATMs i will have been taken out of set $\mathcal{Z}^t$ and added to the other subsets $\mathcal{U}^t, \mathcal{W}^t, \mathcal{X}^t$ and $\mathcal{Y}^t$ when arriving at line 18. The final part of the clustering heuristic in lines 22 – 28 ensures that even more RATMs j which are located closest to RATMs i belonging to $\mathcal{U}^t, \mathcal{W}^t, \mathcal{X}^t$ or $\mathcal{Y}^t$, are added to the perhaps visit subsets. The number of RATMs j added per RATM i is determined by parameter f. It should be noted that RATMs j are only added when they meet an inventory level $\bar{I}_j^t$ which is at least as high as a given percentage g of the holding capacity C_i for a pickup, or lower than a given percentage $1 - g$ for a delivery. The rationale of this final part of the clustering heuristic is to stimulate the exchange of inventories between RATMs i and j to eventually reduce the amount to be picked up from there, and delivered to the depot.

5 Computational Experiments

In this section we present the instances generator in Section 5.1, some implementation details in Section 5.2, and the results of our extensive computational experiments in Section 5.3. The instance set as well as detailed computational results are available at `http://www.leandro-coelho.com/instances`.

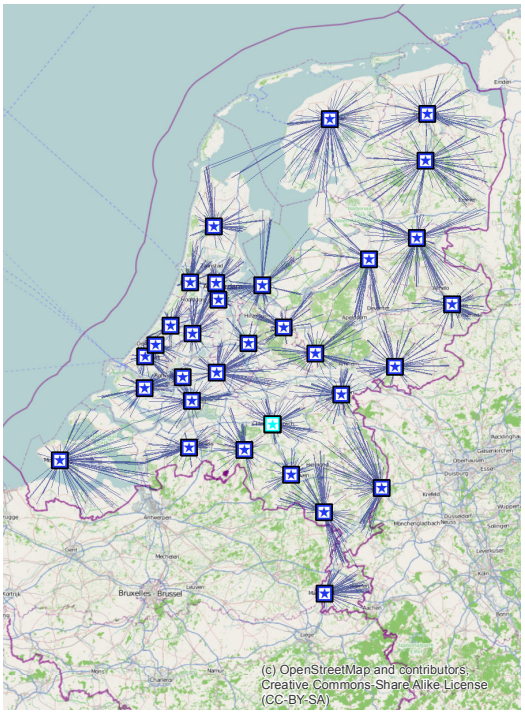
The problem at hand involves RATMs located all over the Netherlands, with over 6,000 vertices and 32 cash centers, each operating with one vehicle. For comparison purposes, the largest instance that can be solved exactly for the IRP, with a single vehicle and a single depot, contains 200 customers [18]. We note that the IRP does not contain many of the features presented in this paper. Likewise, the largest M-M PDP instance solved to optimality contains 200 vertices and a single vehicle [38]. The IRPPD combines features of these two problems and is significantly more complicated than either of these. For these reasons, it is unrealistic to solve this problem exactly for the large sizes we are considering. However, through suitable simplifications, we are able to solve real-world instances and obtain good solutions for this practical case, as we will now show.

5.1 Instances generation

The data used in our experiments stem from a real-world case in the Netherlands. A total of 6,377 cash dispensing self-service devices, both regular and recirculation ATMs, were installed in the Netherlands in 2013 [8]. From a handful of regional distribution centres, cash is transported to and from cross-dock centres, from which the last-mile distribution takes place. Cross-dock centres usually serve the cash supply and demand for a small subregion of the Netherlands. We depict this case in Figure 1.

For our experiments we assume that all self-service devices in the Netherlands are RATMs. This is not unrealistic given the recent strong increase in the share of RATMs in the Netherlands [27]. Also, in several other countries, such as Japan, RATMs already domi-

Figure 1: Map of the Netherlands depicting 32 subregions

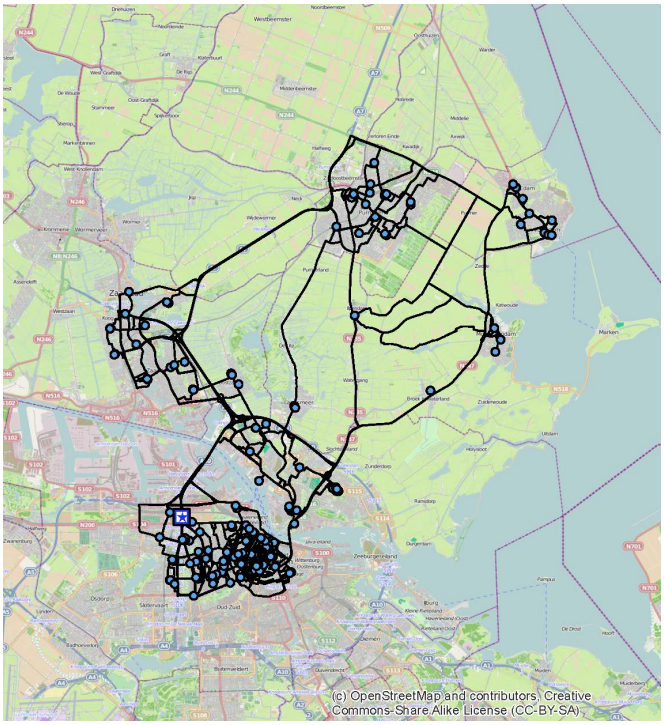


nate the cash self-service device market [46]. To mimic reality we have divided all RATM locations into 32 subregions, each served by a single cross-dock centre, i.e., a depot, each with a maximum of 200 RATMs in order to obtain instances of similar sizes. Although the Netherlands is rather densely populated, not all parts of it are equally well served by RATMs. Their coverage varies considerably over the country. Our solution methodology is rather robust and can easily deal with this diversity. Figure 2 depicts the position of the RATMs and the cash center for the Amsterdam area.

Several experiments were performed with various clustering settings. The parameter settings were gathered from cash supply chain parties in the Netherlands, and estimated when these could not be made publicly available for security reasons. The parameters for the clustering heuristic are set as follows:

- $m \in \{0, 1, 2\}$
- $f \in \{0, 1, 2\}$

Figure 2: Map of the Amsterdam subregion



- $g \in \{0.3, 0.5, 0.7\}$
- $b = \{15\}$.

The parameters for the instance generation are the following:

- $\alpha_i =$ €0.08 per €1,000 inventory per period t (based on a 3% annual interest rate);
- $C_i =$ €260,000 per RATM i. This is an estimation based on an RATM with four cassettes, each with a capacity of 2,000 notes;
- $\beta =$ €0.30 per €1,000 picked up or delivered at the depot;
- $p =$ six periods. Periods coincide with days, which together define a workweek from Monday to Saturday. In practice, order lead-times are not more than one or two days and so a planning horizon of six days is sufficient;
- $\mathcal{K} =$ one vehicle;

- $Q_k =$ €7,800,000 per vehicle k. This is an estimation based on 30 full replenishments of €260,000 in a single period;

- $\gamma_k =$ €2,000 per vehicle k per period used;

- $r = 18$ minutes.

- $S =$ eight hours. This is the regular daily work time in the Netherlands. We assume the vehicle is loaded prior to performing the route, so the driver has at most eight hours to perform all pickups and deliveries;

- $\delta =$ € 8.00 per minute, which is approximately twice as expensive as the regular vehicle renting cost;

- $I_i^0 \in \{0, \ldots, C_i\}$. Each RATM i is assigned a random initial inventory in euros;

- $d_i^t = \{\pm -45,000, \ldots, \pm 45,000\}$ per period. Random values are drawn from a Poisson distribution for both withdrawals and deposits, which are thereafter combined into a net demand. To simulate real demands at different locations, we have ensured that RATMs either have a net-positive, a net-negative or a balanced demand without losing stochasticity. We have then generated different demands at different periods: in periods $t \in \{1, 2\}$ the demand tends to be net-positive, because in the Netherlands more cash is deposited on Mondays and Tuesdays. The demand in periods $t \in \{5, 6\}$ tends to be net-negative, since more cash is withdrawn on Fridays and Saturdays;

- $c_{ij} =$ the arc set is constructed using real travel distances between the RATMs and the depot. A routable network dataset for the Netherlands was constructed using OpenStreetMap data [33] and average driving speeds were used on the various road types to approximate true speeds.

5.2 Implementation features

The algorithm just described was coded in C++ using the IBM Concert Technology and solved with the CPLEX 12.5.1 solver running on a single thread. 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. A time limit of six hours was imposed on the execution of each instance.

Algorithm 1 can be used to optimally solve small to medium instances of the problem. If the instance size is small, even subtour elimination constraints (5) can be fully enumerated. However, for the instances that we consider in this paper, which contain up to 200 nodes, this approach is infeasible. Moreover, some constraints are extremely numerous, e.g., (22) and (26). These two sets of constraints account for almost half a million rows in the LP and their inclusion in the root node of the branch-and-cut solver cause the simplex algorithm to perform poorly when optimizing the linear program relaxation.

In order to cope with this situation, we have decided to add two new layers to the branch-and-cut algorithm. In the first one, we verify at every node having a fractional solution whether inequalities (26) are violated and we then add them to the LP. This helps improve the lower bound of the problem. The second new layer is added to handle constraints (22) since we have observed that these constraints are not tight with respect to the vehicle capacity and are generally satisfied. For this reason, we have devised an algorithm to verify whether they are violated only when at an integer solution. If the integer solution is found to violate constraints (22), these are added and the node is then reoptimized. Otherwise, the solution is feasible for the IRPPD. The procedure created to check whether an integer solution respects constraints (22) is outlined in Algorithm 3. These two changes have significantly decreased the time required to solve the subproblems.

Moreover, we have also observed that the separation algorithm of constraints (5) can be improved by adding a new set of variables representing the route of the vehicle in an undirected graph. By doing this, the separation algorithm runs on a graph half the size

Algorithm 3 Separation algorithm for constraints (22)

Let Vx, Vu, Vp, and Vq be the values of the variables x, u, p, and q, respectively, in an integer solution.
for all $k \in \mathcal{K}$ **do**
  for all $t \in \mathcal{T}$ **do**
    $next \leftarrow 0$.
    for all $j \in \mathcal{V}'$ **do**
      $next+ = jVx^{kt}_{0j}$.
    end for
    if $Vu^{kt}_{next} - Vu^{kt}_{0} - Vp^{kt}_{next} + Vq^{kt}_{next} < 0$ **then**
      Add the violated constraint to the problem:
      $u^{kt}_{next} - u^{kt}_{0} - p^{kt}_{next} + q^{kt}_{next} + (1 - x^{kt}_{0,next})Q_k \geq 0$.
    end if
    $previous = next$.
    while $next \neq 0$ **do**
      $next \leftarrow 0$.
      for all $j \in \mathcal{V}$ **do**
        $next+ = jVx^{kt}_{previous,j}$.
      end for
      if $Vu^{kt}_{next} - Vu^{kt}_{previous} - Vp^{kt}_{next} + Vq^{kt}_{next} < 0$ **then**
        Add the violated constraint to the problem:
        $u^{kt}_{next} - u^{kt}_{previous} - p^{kt}_{next} + q^{kt}_{next} + (1 - x^{kt}_{previous,next})Q_k \geq 0$
      end if
      $previous = next$.
    end while
  end for
end for

of the original one, looking for connected components and deriving maximum cuts over a much smaller network. Moreover, each subtour elimination cut derived for this new variable is equivalent of two cuts expressed in the original variables, one in each direction.

These two procedures employed to generate new cuts dynamically typically yield around 20 thousand cuts only at the root node of the tree. This number, although sizeable, is only a small fraction of the total number of cuts that could potentially be generated. By optimizing a problem with fewer constraints, we gain in speed since far fewer simplex iterations are needed. We note that a typical instance has around half a million binary variables and 150 thousand constraints after the two procedures just described have been applied.

We have also observed that at the beginning of the optimization process, the problem is degenerate, i.e., several pivoting operations do not improve the value of the objective function. We have therefore taken advantage of the fact that during the optimization process, one can change the routing decisions while remaining feasible. Since routing variables x do not appear in the objective function, this different solution does not change the value of the objective function. Note that as long as the total route length is less than the shift duration, the solution remains feasible. We have tested adding a small coefficient to the objective function to further minimize route length, and thus avoid degeneracy, but this option did not yield any significant result.

Finally, in order to obtain a clear view of the performance of our algorithms, and also to speed up the solution of each node, we have turned off the CPLEX cut generation.

5.3 Results of computational experiments

We start our analysis by presenting the results obtained by the algorithm described in Sections 3 and 5.2 on the set of 32 instances described in Section 5.1. These experiments have shown that our algorithm is rather stable, especially when considering the large scale of the instances involving up to 200 vertices. However, the optimality gap is rather large,

with an average of 51%, a maximum of 62% and a minimum of 45%. To benchmark the quality of these solutions, we can compare to the instances of the IRP without pickup and delivery which has recently been solved to optimality in Coelho and Laporte [18] for comparable instance sizes. The problem at hand is much more complicated and clearly requires an additional effort in order to obtain good solutions.

We have then limited the flexibility of the algorithm by disallowing visits to RATMs that do not require a pickup or a delivery to remain operational. This is achieved by setting to zero both parameters m and f of the clustering procedure. Obviously, all solutions obtained for this constrained version of the problem remain valid for the general case, but the lower bounds can no longer be directly compared. In this situation, we have observed an average improvement of 24% in the upper bounds. Under this scenario, seven instances were solved to optimality, and most of them yielded gaps below 0.1%. This is remarkable given the difficulty of the problem which combines characteristics of the IRP and of the M-M PDP. These results are displayed in Table 1, where we present in columns "Upper Bound (general)" and "Upper Bound (clustering)" the obtained upper bounds before and after applying the clustering phase. Again, note that the latter solutions are valid for the general problem. The size of the vertex set $\mathcal{V}'$ is also listed: all instances but two contain 200 RATMs. These two exceptions resulted from the indivisible number of RATMs (i.e., 6,377) over 32 instances. In column "Lower Bound (clustering)" we present the lower bound obtained when solving the problem with the clustering aggregation, i.e., the constrained problem. We observe that these lower bounds are not valid for the general problem. The column "Gap (%)" refers to the optimality gap between the upper and lower bounds computed for the clustering strategy. By increasing the number of periods m in which the algorithm can decide when to serve the RATMs, we obtain a problem that turns out to be very similar to the original one, and the same solutions are obtained when $m = 1$ and $m = 2$.

We have also tested different cases by allowing the algorithm to visit RATMs that do not necessarily need a visit, but that could help reallocate cash throughout the system, thus

Table 1: Full results of the branch-and-cut algorithm with the clustering procedure set with $m = 0$ and $f = 0$

Instance	$\|\mathcal{V}'\|$	Upper Bound (general)	Upper Bound (clustering)	Lower Bound (clustering)	Gap (%)
Inst-01	200	37289	25698	25642	0.22
Inst-02	191	34076	23890	23849	0.17
Inst-03	200	35898	25594	25576	0.06
Inst-04	200	33962	24988	24949	0.15
Inst-05	200	30505	24754	24754	0.00
Inst-06	200	34214	25420	25394	0.10
Inst-07	200	33984	24978	24963	0.06
Inst-08	200	31837	25067	25053	0.05
Inst-09	200	43820	25190	25181	0.03
Inst-10	200	31955	25070	25070	0.00
Inst-11	200	39339	24647	24640	0.02
Inst-12	200	29865	24313	24301	0.04
Inst-13	200	32564	23663	23644	0.08
Inst-14	200	28123	26498	26498	0.00
Inst-15	200	30684	25705	25705	0.00
Inst-16	178	31620	24255	24165	0.37
Inst-17	200	33619	24831	24816	0.06
Inst-18	200	37025	24502	24457	0.18
Inst-19	200	28200	24708	24685	0.09
Inst-20	200	31525	24751	24737	0.05
Inst-21	200	33028	25343	25313	0.12
Inst-22	200	31669	24732	24685	0.19
Inst-23	200	29385	24751	24737	0.05
Inst-24	200	36754	25012	24960	0.20
Inst-25	200	31767	24156	24134	0.09
Inst-26	200	31970	24743	24743	0.00
Inst-27	200	28714	23771	23763	0.03
Inst-28	200	33452	25343	25343	0.00
Inst-29	200	35921	25852	25852	0.00
Inst-30	200	30078	24529	24525	0.01
Inst-31	200	32503	24081	24030	0.21
Inst-32	200	33661	25339	25313	0.10
Average	199	33094	25114	24859	0.09

avoiding the need to return cash to the depot, which incurs a cost. Whenever we set the number f of close-by RATMs allowed to be visited to 1 or 2, but did not allow them to be visited in different periods, i.e., $m = 0$, the same solutions from the general case were obtained once again. This remained true irrespective of the value of the parameter b, i.e., the minimum inventory level of the extra RATMs.

Fixing the number f of extra RATMs and the number m extra of periods, and testing the effect of selecting RATMs based on their minimum average inventory level, i.e., $b = 30$, 50, and 70%, we have observed a clear trend: when the RATMs with higher inventory levels are allowed to be visited, better solutions are obtained. One possible explanation for this is that cash could be moved from these high inventory RATMs to the ones with shortages, hence decreasing both inventory costs and depot handling costs. Table 2 shows a summary of these results.

Table 2: Summary of the results when changing the parameter b, controlling the inventory levels of extra RATMs

Clustering case		Upper Bound (general)	Upper Bound (clustering)	Improvement over the general case (%)
$f = 1$, $m = 1$	$b = 30$	33094	31937	3.51
	$b = 50$	33094	30802	6.94
	$b = 70$	33094	27829	15.79
$f = 1$, $m = 2$	$b = 30$	33094	31949	3.47
	$b = 50$	33094	31944	3.49
	$b = 70$	33094	30840	6.80
$f = 2$, $m = 1$	$b = 30$	33094	31958	3.45
	$b = 50$	33094	31934	3.51
	$b = 70$	33094	31562	4.63
$f = 2$, $m = 2$	$b = 30$	33094	31952	3.46
	$b = 50$	33094	31941	3.49
	$b = 70$	33094	31926	3.54

Finally, since all upper bounds obtained by different clustering procedures remain feasible

for the general case, we are able to compile the best known upper bounds for the problem, obtained from any of the cases presented. These best known upper bounds are presented in Table 3, along with the improvement with respect to the upper bounds obtained for the general case. We observe that solving the problem with our clustering procedure yields better upper bounds for all instances, with average improvements of 29.94% and attaining 47.17% in one case.

6 Conclusions

We have introduced, modelled and solved an inventory-routing problem with pickups and deliveries. We have been succesful in solving a difficult application of the problem arising in the optimization of distribution and inventory management of cash in recirculation ATMs in the Netherlands. The problem was first modelled as a non-linear mixed-integer programming formulation. After linearizing some of the constraints, we have developed an exact branch-and-cut algorithm which was then further strengthened by the inclusion of valid inequalities and some implementation schemes. This algorithm yields initial bounds for the problem, but these are not necessarily tight. A clustering heuristic was developed in order to reduce the problem size, which was then solved again by the branch-and-cut algorithm. Different settings of the clustering procedure were tested, and we have shown which ones are able to provide a good trade-off in terms of simplification of the problem and upper bound values. We were able to solve exactly or to near optimality 32 instances involving up to 200 vertices. This size is similar to that of the largest IRP or M-M PDP instances that have been solved in the past. Our problem is obviously more difficult, because it combines these two features. Since recirculation ATMs are expected to become a market standard in the near future, our paper anticipates on this development by providing cash supply chain parties the means to immediately improve the replenishment operations and yield significant cost savings.

Table 3: Best results of the branch-and-cut algorithm with the clustering procedure

Instance	$\|\mathcal{V}'\|$	Upper Bound (general)	Upper Bound (clustering)	Improvement over the general case (%)
Inst-01	200	37289	22538	39.56
Inst-02	191	34076	23890	29.89
Inst-03	200	35898	25594	28.70
Inst-04	200	33962	24988	26.42
Inst-05	200	30505	24754	18.85
Inst-06	200	34214	25420	25.70
Inst-07	200	33984	24978	26.50
Inst-08	200	31837	25067	21.26
Inst-09	200	43820	25190	42.51
Inst-10	200	31955	25070	21.55
Inst-11	200	39339	24647	37.35
Inst-12	200	29865	24313	18.59
Inst-13	200	32564	23663	27.33
Inst-14	200	28123	26498	5.77
Inst-15	200	30684	25705	16.22
Inst-16	178	31620	18607	41.15
Inst-17	200	33619	21677	35.52
Inst-18	200	37025	22235	39.95
Inst-19	200	28200	24708	12.38
Inst-20	200	31525	19204	39.08
Inst-21	200	33028	18861	42.89
Inst-22	200	31669	24732	21.90
Inst-23	200	29385	19238	34.53
Inst-24	200	36754	19417	47.17
Inst-25	200	31767	19271	39.34
Inst-26	200	31970	19721	38.31
Inst-27	200	28714	23771	17.21
Inst-28	200	33452	20028	40.13
Inst-29	200	35921	21381	40.48
Inst-30	200	30078	24529	18.45
Inst-31	200	32503	24081	25.91
Inst-32	200	33661	21010	37.58
Average	199	33094	23196	29.94

References

[1] 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.

[2] E. Angelelli and R. Mansini. The vehicle routing problem with time windows and simultaneous pick-up and delivery. In A. Klose, M. G. Speranza, and L. N. Van Wassenhove, editors, *Quantitative Approaches to Distribution Logistics and Supply Chain Management*, volume 519 of *Lecture Notes in Economics and Mathematical Systems*, pages 249–267. Springer-Verlag, Berlin, 2002.

[3] S. Anily and R. Hassin. The swapping problem. *Networks*, 22(4):419–433, 1992.

[4] S. Anily, M. Gendreau, and G. Laporte. The preemptive swapping problem on a tree. *Networks*, 58 (2):83–94, 2011.

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

[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] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi. An exact algorithm for the traveling salesman problem with deliveries and collections. *Networks*, 42(1):26–41, 2003.

[8] ABN AMRO Bank, ING Bank, and Rabobank. Automated teller machine locators. online, 2013. URL `http://www.ing.nl/,http://www.abnamro.nl/,http://www.rabobank.nl/`.

[9] M. Benchimol, P. Benchimol, B. Chappert, A. De La Taille, F. Laroche, F. Meunier, and L. Robinet. Balancing the stations of a self-service bike hire system. *RAIRO-Operations Research*, 45(1):37–61, 2011.

[10] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems: a classification scheme and survey. *TOP*, 15(1):1–31, 2007.

[11] N. Bianchessi and G. Righini. Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Computers & Operations Research*, 34(2):578–594, 2007.

[12] D. Chemla, F. Meunier, and R. Wolfler Calvo. Bike sharing systems: solving the static rebalancing problem. *Discrete Optimization*, 10(2):120–146, 2013.

[13] M. Christiansen, K. Fagerholt, and D. Ronen. Ship routing and scheduling: Status and perspectives. *Transportation Science*, 38(1):1–18, 2004.

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

[15] M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen. Ship routing and scheduling in the new millennium. *European Journal of Operational Research*, 228(3):467–483, 2013.

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

[17] 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*, Forthcoming, 2013.

[18] L. C. Coelho and G. Laporte. Improved solutions for inventory-routing problems through valid inequalities and input ordering. Technical Report CIRRELT-2013-33, Montreal, 2013.

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

[20] L. C. Coelho, J.-F. Cordeau, and G. Laporte. Thirty years of inventory-routing. *Transportation Science*, Forthcoming, 2014.

[21] C. Contardo, C. Morency, and L.-M. Rousseau. Balancing a dynamic public bike-sharing system. Technical Report CIRRELT-2012-09, Montreal, 2012.

[22] J.-F. Cordeau and G. Laporte. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153(1):29–46, 2007.

[23] G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2(4):393–410, 1954.

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

[25] M. Dell'Amico, G. Righini, and M. Salani. A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, 40(2):235–247, 2006.

[26] G. Desaulniers, J. Desrosiers, A. Erdmann, M. M. Solomon, and F. Soumis. Vehicle routing problem with pickup and delivery. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *Monographs on Discrete Mathematics and Applications*, pages 225–242. SIAM, Philadelphia, 2002.

[27] ECB. ATM cash deposits at terminals located in the country with cards issued in the country. Technical Report DA12, European Central Bank, 2013. URL `http://sdw.ecb.europa.eu/`.

[28] G. Erdoğan, G. Laporte, and J.-F. Cordeau. A branch-and-cut algorithm for the non-preemptive capacitated swapping problem. *Discrete Applied Mathematics*, 158(15):1599–1614, 2010.

[29] G. Erdoğan, G. Laporte, and R. Wolfler Calvo. The one commodity pickup and delivery traveling salesman problem with demand intervals. Technical Report CIRRELT-2013-46, Montreal, 2013.

[30] G. Erdoğan, M. Battarra, G. Laporte, and D. Vigo. Metaheuristics for the travelingsalesman problem with pickups, deliveries and handling costs. *Computers & Operations Research*, 39(5):1074–1086, 2012.

[31] M. Gendreau, G. Laporte, and F. Semet. The covering tour problem. *Operations Research*, 45(4): 568–576, 1997.

[32] M. Gendreau, G. Laporte, and D. Vigo. Heuristics for the traveling salesman problem with pickup and delivery. *Computers & Operations Research*, 26(7):699–714, 1999.

[33] Geofabrik GmbH and OpenStreetMap Contributors. Openstreetmap netherlands shapefile. online, 2013. URL `http://download.geofabrik.de/europe/netherlands-latest.shp.zip`.

[34] I. Gribkovskaia, G. Laporte, and M. Vlček. General solutions to the single vehicle routing problem with pickups and deliveries. *European Journal of Operational Research*, 180(2):568–584, 2007.

[35] H. Hernández-Pérez and J.-J. Salazar-González. The one-commodity pickup-and-delivery traveling salesman problem. In M. Jünger, G. Reinelt, and G. Rinaldi, editors, *Combinatorial Optimization - Eureka, You Shrink!*, volume 2570 of *Lecture Notes in Computer Science*, pages 89–104. Springer, Berlin, 2003.

[36] H. Hernández-Pérez and J.-J. Salazar-González. A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics*, 145(1):126–139, 2004.

[37] H. Hernández-Pérez and J.-J. Salazar-González. Heuristics for the one-commodity pickup-and-delivery traveling salesman problem. *Transportation Science*, 38(2):245–255, 2004.

[38] H. Hernández-Pérez and J.-J. Salazar-González. The one-commodity pickup-and-delivery traveling salesman problem: Inequalities and algorithms. *Networks*, 50(4):258–272, 2007.

[39] A. Hoff, I. Gribkovskaia, G. Laporte, and A. Løkketangen. Lasso solution strategies for the vehicle routing problem with pickups and deliveries. *European Journal of Operational Research*, 192(3): 755–766, 2009.

[40] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the Association for Computing Machinery*, 7(4):326–329, 1960.

[41] H. Min. The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research Part A: General*, 23(5):377–386, 1989.

[42] G. Mosheiov. The travelling salesman problem with pick-up and delivery. *European Journal of Operational Research*, 79(2):299–310, 1994.

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

[44] J. G. Rakke, M. Stålhane, C. R. Moe, M. Christiansen, H. Andersson, K. Fagerholt, and I. Norstad. A rolling horizon heuristic for creating a liquefied natural gas annual delivery program. *Transportation Research Part C: Emerging Technologies*, 19(5):896–911, 2011.

[45] T. Raviv, M. Tzur, and I. A. Forma. Static repositioning in a bike-sharing system: models and solution approaches. *European Journal on Transportation and Logistics*, 2(3):187–229, 2013.

[46] Retail Banking Research Ltd. RBR. Deposit automation and recycling. Technical report, RBR, London, 2012.

[47] A. Subramanian and M. Battarra. An iterated local search algorithm for the travelling salesman problem with pickups and deliveries. *Journal of the Operational Research Society*, 64(3):402–409, 2013.

[48] A. Subramanian, L. M. A. Drummond, C. Bentes, L. S. Ochi, and R. Farias. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 37(11):1899–1911, 2010.

[49] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40(1):475–489, 2013.

[50] E. E. Zachariadis, C. D. Tarantilis, and C. T. Kiranoudis. An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries. *European Journal of Operational Research*, 202(2):401–411, 2010.

[51] F. Zhao, S. Li, J. Sun, and D. Mei. Genetic algorithm for the one-commodity pickup-and-delivery traveling salesman problem. *Computers & Industrial Engineering*, 56(4):1642–1648, 2009.