CIRRELT

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

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

# Exact Solution Methods for the Multi-Vehicle Multi-Period Vehicle Routing Problem with Due Dates

**Homero Larrain**
**Leandro C. Coelho**
**Claudia Archetti**
**Maria Grazia Speranza**

**January 2018**

**CIRRELT-2018-06**

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2018-003.

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

# Exact Solution Methods for the Multi-Vehicle Multi-Period Vehicle Routing Problem with Due Dates

**Homero Larrain[1,*], Leandro C. Coelho[2], Claudia Archetti[3], Maria Grazia Speranza[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, Université Laval, 2325 de la Terrasse, Québec, Canada G1V 0A6

[3] Department of Economics and Management, University of Brescia, Via S. Faustino 74/b-25122 Brescia, Italy

**Abstract.** In this paper we study the vehicle routing problem with due dates.In this problem, a supplier has to determine a distribution plan to visit a set of customers over a given planning horizon. Each customer is associated with a release date and a due date, corresponding to the date at which the goods required by the customer become available at the supplier's depot, and the date by which the customer has to be visited. A fleet of capacitated vehicles is available at the depot to perform the distribution and the objective is to minimize the distribution costs and the cost related to delayed deliveries. A variable MIP neighborhood descent (VMND) algorithm is proposed to solve the problem, which speeds up the search for high quality solutions through a local search heuristic embedded in an branch-and-bound scheme. Computational tests on benchmark instances are performed to evaluate the VMND algorithm against a known branch-and-cut method. The results show that our algorithm improves 44 out of 90 best known solutions on benchmark instances where one vehicle is available. We also observe that the VMND algorithm match or beat the solution of a pure branch-and-bound implementation in just one fifth of the time in average. We conclude that the VMND algorithm, which was proposed in the literature for a different problem, is easily adaptable to other problems.

**Keywords**. Routing, due dates, heuristic, local search, exact algorithm, hybrid algorithm, VMND.

**Acknowledgements.** 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.puc.cl

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2018

# 1 Introduction

Distribution problems involve various levels of decisions including operational ones that depend on short term information. A class of classical operational problems arising in distribution are Vehicle Routing Problems (VRPs), for which the aim is to determine vehicle routes for the distribution. Traditionally, in the literature, VRPs are concerned with decisions over a single period [18] where the demand of customers is known. The exception has been the periodic VRP (PVRP), in which one aims to create routes over multiple periods, typically days, to be repeated over a longer planning horizon [17]. In the PVRP, one needs to choose, for each customer, the visiting pattern, that is the visiting periods, from a set of available options, and then create vehicle routes visiting customers on the selected periods. Extensions of the PVRP have been discussed in Francis et al. [16] and Archetti et al. [6].

When dealing with multi-period distribution, an important class of problems studied in the literature is the Inventory Routing Problem (IRP), which combines routing and inventory management. In the IRP one aims to satisfy customer demands over several periods while minimizing the sum of distribution and inventory costs, or the distribution costs only [8, 9, 14]. A central decision maker is assumed to be in charge of all routing and inventory decisions, as long as customer capacities are respected and their consumption is satisfied. This management policy is called vendor-managed inventory (VMI), as all decisions are centralized to the vendor. IRPs have an established body of research with benchmark instances and many competing algorithms such as local search-based heuristics [10, 12], branch-and-cut [1, 3, 11], branch-price-and-cut [15], and hybrid algorithms that combine heuristic and exact methods [2, 5, 13]. The alternative and more traditional management policy is the retailer-managed inventory (RMI) (introduced with this name, and in contrast with the VMI, in Bertazzi and Speranza [7]) in which customers decide when they want to be served (usually a range of time, such as a day, or a window within a day, or a set of possible days). Thus, the supplier is not responsible for the inventory management and is in charge of just organizing the vehicle routes over time.

When the distribution follows an RMI policy, the quantities to be delivered are determined

by the customers and are, thus, known in the models and algorithms. If the day of service is fixed by the customers, then VRPs are the most appropriate tools, possibly with time windows. If customers are more flexible and a range of days for the service is possible, then a customer is in general associated with a delivery release date and a due date, which are the earliest and latest possible service times, respectively. This problem appears in many contexts, such as in retail online shopping. In these cases, customers place an order and can choose a delivery due date among different options which have different shipping fees. When such an order is placed, the shipper has to deliver the product within the due date. In the multi-period vehicle routing problem with due dates (MVRPD) introduced in Archetti et al. [4], each customer must be served between a release and a due date. A fleet of capacitated vehicles is available at the depot to perform the distribution and the objective is to minimize the distribution cost. In Archetti et al. [4] different formulations of the problem are proposed and compared. The conclusion is that the best one is a load-based formulation. Computational experiments show that instances with up to 35 customers can be systematically solved to optimality over a time horizon of six days and up to three days of flexibility.

In this paper the MVRPD is tackled and a new exact method is proposed which speeds up the search for high quality solutions through a local search heuristic embedded in an exact scheme. The goal is to increase the maximum size of instances solved to optimality and, when optimality cannot be achieved, reduce the optimality gaps and find high quality solutions. The algorithm relies on a heuristic that makes use of neighborhoods defined through constraints that are added to the original model allowing to solve it quickly and efficiently. This algorithmic scheme, called Variable MIP Neighborhood Descent (VMND), was introduced by Larrain et al. [19] and successfully applied to a cash logistics problem. The algorithm has been empirically shown to significantly outperform a branch-and-cut algorithm using less than 40% of the running time. We derive new neighborhoods tailored for the MVRPD and compare the performance of the VMND against the original branch-and-bound algorithm of Archetti et al. [4] and an improved version introduced in this paper.

The remainder of this paper is organized as follows. In Section 2 we provide a formal definition and a mathematical programming formulation for the MVRPD. The description of the VMND is given in Section 3. Computational experiments are presented in Section 4, followed by our conclusions and findings in Section 5.

# 2 Problem description and mathematical model

In this section, we first recall the MVRPD and the load-based formulation presented in Archetti et al. [4]. Then, we describe families of existing and new valid inequalities.

The MVRPD is defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$. The set of nodes $\mathcal{V} = \{0, ..., V\}$ includes the depot, represented by node 0, and $V$ customers denoted by set $\mathcal{V}' = \{1, ..., V\}$. The set of arcs $\mathcal{A}$ contains every pair of nodes $i, j \in \mathcal{V}$ with $i \neq j$. A fleet of $m$ homogeneous vehicles of capacity $Q$ is available to perform the distribution. Each arc $(i, j) \in \mathcal{A}$ has an associated routing cost of $c_{ij}$ which is incurred if the arc is traversed by a vehicle. A planning horizon $\mathcal{H} = \{1, ..., H\}$ of $H$ periods is considered. Each customer $i \in \mathcal{V}'$ places an order that occupies $q_i$ units of the vehicle capacity. This order is made available at the depot on its release date $r_i \in \mathcal{H}$, and has to be delivered to the customer not later than its due date, $d_i$. There are two types of customers: mandatory and optional. Mandatory customers are the ones associated with a due date $d_i \in \mathcal{H}$, i.e., they have to be served within the end of the planning horizon. Optional customers, instead, are associated with a due date which is greater than $H$. Thus, their service can be postponed, i.e., it is possible to deliver the corresponding order after the end of the planning horizon. The set of optional customers is denoted as $\mathcal{C}$. A cost $p_i$ is paid when postponing customer $i \in \mathcal{C}$. Every period that an order stays at the depot after its release date incurs a handling cost of $h_i$.

The formulation is based on the following variables. First, a set of binary routing variables $X = \{x_{ij}^t\}, (i, j) \in \mathcal{A}, t \in \mathcal{H}$ are needed where $\{x_{ij}^t\}$ takes value 1 when arc $(i, j)$ is traversed by a vehicle in period $t$. We set to 0 all routing variables associated with periods that are outside the customers service window, i.e., $x_{ij}^t := 0, \forall i, j \in \mathcal{V}, t \in \mathcal{H} : t <$

$r_i \vee t > d_i$. A set of non-negative continuous load variables $L = \{l_{ij}^t\}, (i,j) \in \mathcal{A}, t \in \mathcal{H}$ is also defined, where variable $l_{ij}^t$ indicates the load carried by a vehicle when traversing arc $(i,j)$ in period $t$.

The MPVRPD is formulated as a mixed integer programming problem that minimizes a total cost function defined as:

$$\sum_{(i,j)\in\mathcal{A}} \sum_{t\in[r_i,H]} c_{ij}x_{ij}^t + \sum_{i\in\mathcal{V}} h_i \sum_{t\in[r_i,H]} (t-r_i) \sum_{j\in\mathcal{V}} x_{ij}^t + \sum_{i\in\mathcal{C}} (h_i(H-r_i)+p_i) \left(1 - \sum_{j\in\mathcal{V}} \sum_{t\in[r_i,H]} x_{ij}^t\right). \tag{1}$$

Function (1) is the sum of three cost components: the routing costs, the inventory holding cost for customers visited during the planning horizon, and the penalty and inventory holding costs for postponed customers.

Cost function (1) has to be minimized subject to the following constraints:

$$\sum_{j\in\mathcal{V}} \sum_{t\in[r_i,d_i]} x_{ij}^t = 1, \quad i \in \mathcal{V}' \setminus \mathcal{C}, \tag{2}$$

$$\sum_{j\in\mathcal{V}} \sum_{t\in[r_i,d_i]} x_{ij}^t \leq 1, \quad i \in \mathcal{C}, \tag{3}$$

$$\sum_{j\in\mathcal{V}} x_{ij}^t = \sum_{j\in\mathcal{V}} x_{ji}^t, \quad i \in \mathcal{V}, t \in \mathcal{H}, \tag{4}$$

$$\sum_{j\in\mathcal{V}} x_{0j}^t \leq m, \quad t \in \mathcal{H}, \tag{5}$$

$$\sum_{j\in\mathcal{V}} l_{ji}^t - \sum_{j\in\mathcal{V}} l_{ij}^t = q_i \sum_{j\in\mathcal{V}} x_{ij}^t, \quad i \in \mathcal{V}', t \in \mathcal{H}, \tag{6}$$

$$\sum_{j\in\mathcal{V}} l_{0j}^t - \sum_{j\in\mathcal{V}} l_{j0}^t = \sum_{(i,j)\in\mathcal{A}} q_i x_{ij}^t, \quad t \in \mathcal{H}, \tag{7}$$

$$l_{ij}^t \leq Q x_{ij}^t, \quad (i,j) \in \mathcal{A}, t \in \mathcal{H}, \tag{8}$$

$$x_{ij}^t \in \{0,1\}, \quad (i,j) \in \mathcal{A}, t \in \mathcal{H}, \tag{9}$$

$$l_{ij}^t \geq 0, \quad (i,j) \in \mathcal{A}, t \in \mathcal{H}. \tag{10}$$

Constraints (2) state that every mandatory customer is visited once during the delivery window, while constraints (3) ensure that optional customers are visited at most once. Constraints (4) are flow conservation on traversal of the arcs. Constraints (5) limit the number of vehicles to the fleet size. Constraints (6) and (7) link load and flow variables and ensure load conservation. Constraints (8) establish that the loads do not exceed vehicle capacity, and constraints (9) and (10) define the domain of the variables.

The following valid inequalities were proposed in Archetti et al. [4] to tighten the load based formulation. Set $\mathcal{S}_{t_1,t_2}$ denotes the customers that must be served within interval $[t_1, t_2]$, i.e., $\mathcal{S}_{t_1,t_2} := \mathcal{V}' : t_1 \leq r_i \wedge d_i \leq t_2$. Parameter $q_{t_1,t_2}$ represents the total load that must be delivered to customers in $\mathcal{S}_{t_1,t_2}$, i.e., $q_{t_1,t_2} := \sum_{i \in \mathcal{S}_{t_1,t_2}} q_i$.

$$\sum_{(i,j)\in\mathcal{A}} \sum_{t\in[1,t']} q_i x_{ij}^t \leq mQt', \quad t' \in \mathcal{H}, \tag{11}$$

$$\sum_{i\in\mathcal{V}} q_i - \sum_{(i,j)\in\mathcal{A}} \sum_{t\in[0,t']} q_i x_{ij}^t - \sum_{i\in\mathcal{C}} q_i \left(1 - \sum_{j\in\mathcal{V}} \sum_{t\in[r_i,H]} x_{ij}^t\right) \leq mQ(H-t'), \quad t' \in \mathcal{H}, \tag{12}$$

$$\sum_{i\in\mathcal{V}} q_i - \sum_{(i,j)\in\mathcal{A}} \sum_{t\in[0,t']} q_i x_{ij}^t - \sum_{i\in\mathcal{C}} q_i \left(1 - \sum_{j\in\mathcal{V}} \sum_{t\in[r_i,H]} x_{ij}^t\right) \leq Q \sum_{t\in[t'+1,H]} \sum_{j\in\mathcal{V}'} x_{0j}^t, \quad t' \in \mathcal{H}, \tag{13}$$

$$\sum_{j\in\mathcal{V}} \sum_{t\in[t_1,t_2]} x_{0j}^t \geq \left\lceil \frac{q_{t_1,t_2}}{Q} \right\rceil, \quad t_1, t_2 \in \mathcal{H}, t_1 \leq t_2, \tag{14}$$

$$\sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{S}_{t_1,t_2}} \sum_{t\in[t_1,t_2]} q_j x_{ij}^t \geq q_{t_1,t_2}, \quad t_1, t_2 \in \mathcal{H}, t_1 \leq t_2, \tag{15}$$

$$\sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{S}_{t_1,t_2}} \sum_{t\in[t_1,t_2]} q_j x_{ij}^t \leq mQ(t_2 - t_1 + 1), \quad t_1, t_2 \in \mathcal{H}, t_1 \leq t_2. \tag{16}$$

Constraints (11) state that the total delivered load between time periods 1 and $t'$ is bounded by the total capacity of the $m$ vehicles available during the $t'$ periods. Inequalities (12) state that the load delivered between periods $t'+1$ and $H$ does not exceed the capacity of the fleet during the $H - t'$ periods. Constraints (13) tighten the bound on the vehicle capacity by considering only the subset of the vehicles that leave the depot on each time period. Valid inequalities (14) set a lower bound on the number of routes between periods $t_1$ and $t_2$. Constraints (15) ensure that the total load delivered within $[t_1, t_2]$ meets the

aggregated demand for this interval, and (16) limit this total load to the capacity of the fleet in $[t_1, t_2]$.

We now introduce the following two new valid cuts to the formulation:

$$\sum_{(i,j)\in\mathcal{A}} \sum_{t\in[1,t']} q_i x_{ij}^t \leq Q \sum_{t\in[1,t']} \sum_{j\in\mathcal{V}'} x_{0j}^t, \quad t' \in \mathcal{H}, \tag{17}$$

$$\sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{S}_{t_1,t_2}} \sum_{t\in[t_1,t_2]} q_j x_{ij}^t \leq Q \sum_{t\in[t_1,t_2]} \sum_{j\in\mathcal{V}'} x_{0j}^t, \quad t_1, t_2 \in \mathcal{H}, t_1 \leq t_2. \tag{18}$$

These cuts tighten (11) and (16) by considering just the vehicles actually used, as in (13).

# 3 Variable MIP Neighbordhood Descent Algorithm

In this section we describe the algorithm used to solve the MVRPD. We first provide an overview of the VMND algorithm in Section 3.1. Then, in the following two sections, we present the exact solution scheme and the local search procedures used to speed up the search for high quality solutions. Finally, in Section 3.4 the description of the VMND algorithm is completed with the initialization phase.

## 3.1 Overview of the VMND algorithm

The main idea of the VMND is the following. The MILP formulation of the problem is solved with an exact solution method. The execution of the exact algorithm is suspended at some specific points and then a local search is performed around the current solution to speed up the search for high quality solutions. When the local search terminates, the exact solution algorithm is resumed. The local search explores neighborhoods of the current solution, exploiting the power of mathematical programming solvers by defining neighborhoods as constrained versions of the mathematical programming formulation. More precisely, the value of a subset of variables is fixed to the value they have in the current solution. The subset of variables whose value is fixed is such that the resulting

MILP is easy to solve. Then, it becomes computationally acceptable to explore several different neighborhoods of the same current solution.

The local search is performed whenever a new best solution is found or when a time limit is reached. A set of different neighborhoods is defined. If a neighborhood is completely explored without providing a new best solution, the local search explores a new neighborhood. The local search phase ends and the exact solution algorithm is resumed in one of these two cases: when all neighborhoods have been explored with no improvement to the solution, or when the local search finds a new best solution in a specific neighborhood. The VMND keeps track of the last neighborhood it explored in such a way that, in case the exact solution algorithm does not find a new best solution, it resumes the local search on a new neighborhood to avoid redundant searches. The procedure is illustrated in Figure 1.

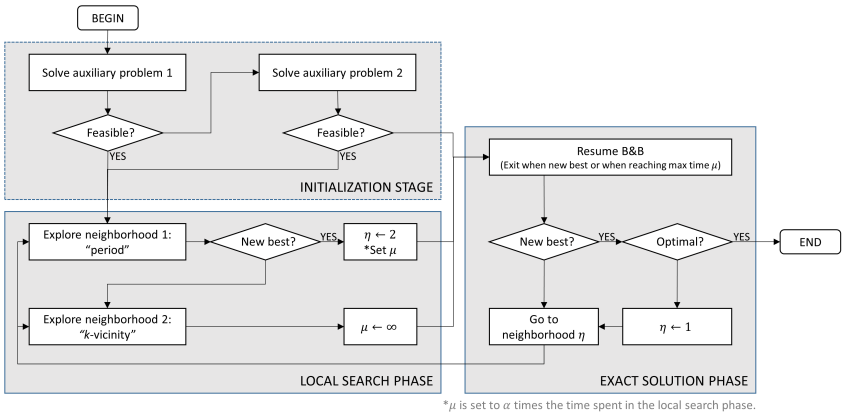


**Figure 1:** Overview of the VMND algorithm for the MVRPD.

In Sections 3.2 and 3.3 we present the exact solution algorithm and the local search in more details, respectively. The initialization phase to find an initial feasible solution is described in Section 3.4.

## 3.2 Exact solution algorithm

The exact solution algorithm used is a branch-and-bound algorithm that solves the MILP defined by the objective function (1), subject to constraints (2) – (10) and a subset of

valid inequalities from (11) – (18).

As Figure 1 illustrates, the execution of the exact algorithm is temporarily interrupted and the local search is performed whenever a new best solution is found, or when a maximum time limit allowed $\mu$ is reached. This value is set as a proportion $\alpha$ of the time spent on the local search phase.

Whenever the exact algorithm finds a new best solution, the local search starts exploring the first neighborhood. If the maximum time limit is reached without a new best solution, the local search is performed on the current solution by exploring a new neighborhood. After the local search is finished, its best solution is fed back to the exact solution phase that keeps track of both the best primal and the dual bounds.

## 3.3 Local search

In the local search phase, neighborhoods are explored to search for improvements on the current solution. A set of neighborhoods $\mathcal{N}$ is defined. Each neighborhood $\eta \in \mathcal{N}$ is associated with a rule to determine the set of variables whose value is fixed in the solution of the MILP. The rule is such that we avoid generating infeasible or too difficult MILPs. To this aim, neighborhoods are defined by fixing the value of subsets of the routing variables $x$ only and letting the variables $l$ free.

Two kinds of neighborhoods, called *period* and *k-vicinity* neighborhoods, have been designed. Both are depicted in Figure 2. The idea behind the *period* neighborhoods is to fix all decisions across the planning horizon except for one period at a time. This does not allow any change on customer delivery dates, but in the period under consideration customers may be assigned to different vehicles and the vehicle routes may be improved. This is done for all periods, one at a time. Given a time period $t_p$, the set of variables to fix for the local search is $\{x_{ij}^t : t \neq t_p\}$, i.e., all the arcs that do not belong to period $t_p$.

In the *k-vicinity* neighborhoods, the idea is to focus the search on a set of customers that form a spatial cluster, allowing changes in the period of visit. This is done by selecting a seed customer $i$ and its $k-1$ closest neighbors in terms of travel costs $c_{ij}$. The set

that includes $i$ and its $k-1$ closest neighbors is defined as the $k$-vicinity of $i$, denoted by $V_k(i)$. When exploring neighborhood $V_k(i)$ these customers are free, i.e., every $x$ variable incident to these customers is left free across all planning periods, while all remaining arcs are fixed. This is done for all possible customers $i_p$. For a given customer $i_p$, the set of variables to fix is $\{x_{ij}^t : i \notin V_k(i_p) \wedge j \notin V_k(i_p)\}$. We set the value of $k$ to 20.

The local search explores the *period* neighborhoods for every possible period $t_p$, and explores the *k-vicinity* neighborhoods for every possible seed customer $i_p$, until no improvement is found. For the *k-vicinity* neighborhoods, customers are cyclically taken as seeds to define the $k$-vicinity. Whenever a new best solution is found, the current best solution is updated for the further exploration of this neighborhood. The algorithm keeps searching the neighborhood and updating the best known solution. The search is stopped when the current best solution is not improved by any of the $k$-vicinity neighborhoods, i.e., when the neighborhoods defined by every possible seed customer $i$ have been explored without improvement. In the case of the *period* neighborhoods, changing the routing in one period has little impact on the solution of the other periods (the only way this can happen is when a optional customer is dropped from or added to a route). Thus, to speed up the exploration of these neighborhoods, we only explore each period once, even if an improving solution has been found.

The maximum time $\mu$ for the exact solution phase is set equal to the time spent in the previous local search multiplied by $\alpha$. Results from Larrain et al. [19] show that a value of $\alpha = 1$ is effective and that the results are not very sensitive to this parameter. When the local search does not find an improving solution, then $\alpha$ is set to infinity, not imposing a time limit to the exact solution phase. In this case, the algorithm will keep working on the exact solution phase until it obtains a new best known solution, or until proving optimality.

Figure 2 provides an illustrative example of the idea of the two kinds of neighborhoods. On the top row the current solution with the routes performed in each time period is shown. The middle row shows what happens when the *period* neighborhood for time period $t_p = 2$ is considered. In particular, the only route performed in period 2 is discarded so that a

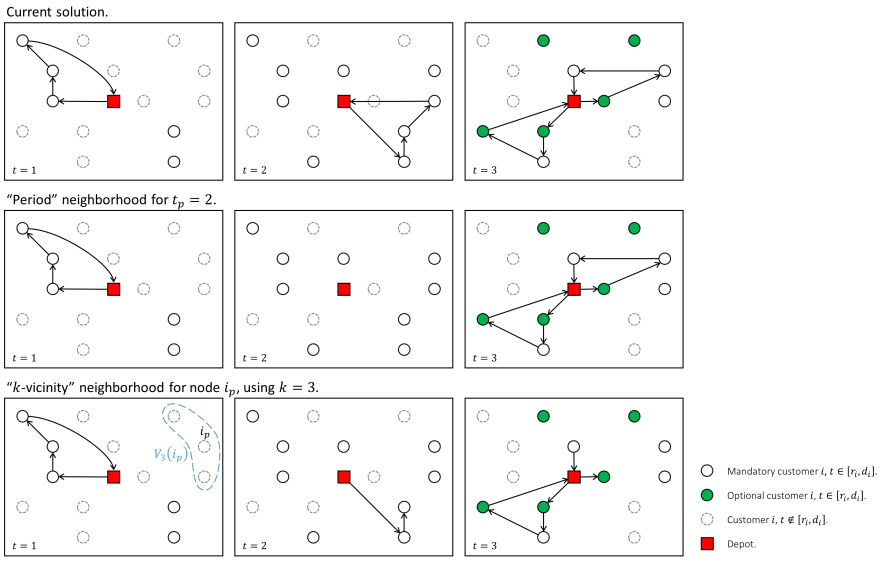


**Figure 2:** Neighborhood definition for the implemented VMND for the MVRPD.

new (and possibly better) route is searched for in the MILP where the $x$ variables related to period 2 are free and the remaining $x$ variables for all other periods are fixed to their value in the current solution. The bottom row shows the $k$-vicinity neighborhoods for node $i_p$ with $k = 3$. The set of customers to explore, $V_3(i_p)$, is depicted in the first box on the left from this row. For these customers, all entering and exiting arcs in all time periods are removed.

## 3.4 Initialization

The branch-and-bound algorithm can take a long time to find a first feasible solution for large instances. To tackle this issue we implemented an initialization procedure which aims at finding a feasible solution to the problem by solving two constrained versions of the original MILP problem. The first auxiliary MILP problem ignores the flexibility allowed by the due dates and serves only the mandatory customers on their release dates. If this first auxiliary problem fails to find a feasible solution, the second one looks for a solution that visits only mandatory customers and allows them to be served at most one day after their release dates. If capacity conditions are not binding, both MILP problems may generate a feasible solution.

If an initial solution is found through these procedures, the VMND starts with the local search phase. If it fails, then the algorithm starts with the exact algorithm with no limit on the computing time allowed (i.e., $\mu = \infty$) to prevent it to exit without a feasible solution for the local search. This process is depicted in Figure 1.

# 4 Computational experiments

In this section we describe the experimental setting adopted to study the performance of the VNMD and the results obtained. The exact algorithm presented in Archetti et al. [4], the one presented in this paper in Section 2, and the VNMD, were all implemented in C++ and run on a machine equipped with a Xeon processor running at 2.77GHz and 96GB of memory, using CPLEX 12.6 as a solver. All experiments were limited to two hours of running time.

We have reimplemented the algorithm of Archetti et al. [4] and rerun all instances for two main reasons. First, there were minor implementation issues in the original paper, affecting the cost structure of the solution (but not the limits of the algorithm); reimplementing and rerunning all instances allowed us to fix these issues and compare solution costs correctly. The second reason is that we wanted to have a clear comparison, running all instances on the same machines, allowing us to compare run times.

In Section 4.1 the benchmark instances and the procedure used to generate new larger ones are described. Section 4.2 is devoted to identifying the most effective combination of cuts and Section 4.3 presents the results of extensive tests performed to study the efficiency of each solution method.

## 4.1 Instances

The experiments were performed on a set of 80 instances, denoted as $(V/H/inv/k)$. An instance is defined by its number of customers, $V := |\mathcal{V}'|$, with $V \in \{25, 50, 75, 100\}$, the number of periods, $H := |\mathcal{H}| \in \{3, 6\}$, and its level of inventory cost, $inv \in \{h, l\}$, where

$h$ and $l$ stand for high or low inventory costs, respectively. Five instances ($k = 1, 2, 3, 4, 5$) were generated for each combination of $V$, $H$ and $inv$. When presenting average results the notation ($V/H/inv$) is used.

Instances of size $V \leq 50$ were taken from the benchmark set of Archetti et al. [4]. Larger instances with $V > 50$ were randomly generated, following the same rules used to build the original set.

## 4.2 Streamlining the B&B implementation

Some preliminary tests were carried out to assess the effectiveness of the B&B algorithm and showed that the entire set of cuts was increasing the computational time more with respect to subsets of them. Thus, to identify a good combination of cuts, instances ($100/3/h/k$) with $k = 1, ..., 5$ were considered and tested. A single class of valid inequalities at a time was first tested (Test 1). Then, the class of valid inequalities that performed best in the first test was chosen and a second set of tests was performed where this class was coupled with each other class (Test 2). Then, we chose the best pair of cuts and a third set of tests was run where the best pair was coupled with each remaining class of cuts (Test 3).

The valid inequalities, detailed in the previous section, are summarized in Table 1.

The following observations helped us reduce the number of tests to perform. First, there are three pairs of cuts that work on the same idea, but using a different measure of fleet capacity: 1a and 1b, 2a and 2b, and 5a and 5b. For instance, deliveries within interval $[1, t']$ should not exceed the total capacity of the fleet in the $t'$ periods. Valid cut 1a (see inequality (11)) expresses this idea by using the total fleet capacity, $mQt'$, as an upper bound, while inequality 1b (see (17)) takes into account the actual number of vehicles leaving the depot, i.e., the used capacity, $Q \sum_{t \in [t_1, t_2]} \sum_{j \in \mathcal{V}'} x_{0j}^t$. Since these constraints are very similar in concept, whenever one of them is included in a pair or a triplet, the other one is not included. The second observation is that cuts 1a and 1b are particular cases of cuts 5a and 5b, with $t_1 = 1$ and $t_2 = t'$. Thus, we decided to leave out 1a and

| Cut | Equation | Description | Interval |
|---|---|---|---|
| 1a | (11) | Deliveries $\leq$ Fleet capacity | $[1, t']$ |
| 1b | (17) | Deliveries $\leq$ Used capacity | $[1, t']$ |
| 2a | (12) | Deliveries $\leq$ Fleet capacity | $[t', H]$ |
| 2b | (13) | Deliveries $\leq$ Used capacity | $[t', H]$ |
| 3 | (14) | Routes $\geq$ Minimum number of routes | $[t_1, t_2]$ |
| 4 | (15) | Deliveries $\geq$ Mandatory demand | $[t_1, t_2]$ |
| 5a | (16) | Deliveries $\leq$ Fleet capacity | $[t_1, t_2]$ |
| 5b | (18) | Deliveries $\leq$ Used capacity | $[t_1, t_2]$ |

**Table 1:** Summary of valid cuts for the MVRPD

1b from this analysis. Although a similar argument could be used to eliminate 2a and 2b from this experiment, we decided to keep them because the total delivery quantity is computed differently (see (12) and (13)).

The results of this experiment are summarized in Table 2. For each test, the best upper bound obtained within two hours of computing time is reported. We focus on the value of the upper bound as we aim at finding the best formulation to be used in the VMND algorithm for which the main goal is to provide high quality solutions. The results show that cuts 2a and 3 are on average more effective than other pairs. These are the cuts that we use in the following tests for the B&B and the VMND algorithms. These results also show that, in most cases, cuts considering the total available fleet as an upper bound of capacity work better than the cuts that consider the used capacity instead.

## 4.3 Computational results

The results obtained for each algorithm are reported in this section. We start by comparing the performance of the B&B proposed in Archetti et al. [4] (B&B-1 from now on) with the results of the B&B proposed in this paper (B&B-2), that makes use of the valid cuts identified in the previous section. In Table 3 the instance description is followed by

| Test 1 | | Test 2 | | Test 3 | |
|---|---|---|---|---|---|
| Cuts | UB | Cuts | UB | Cuts | UB |
| - | 20508.3 | **3, 2a** | **14743.5** | 2a, 3, 4 | 15389.8 |
| 2a | 15690.5 | 3, 2b | 26621.7 | 2a, 3, 5a | 15283.9 |
| 2b | 18785.9 | 3, 4 | 15585.9 | 2a, 3, 5b | 15348.0 |
| **3** | **15132.1** | 3, 5a | 19020.0 | | |
| 4 | 17847.6 | 3, 5b | 18994.5 | | |
| 5a | 17807.5 | | | | |
| 5b | 19836.5 | | | | |
| Add cut (14) | | Add cut (12) | | Stop | |

**Table 2:** Results of tests on valid inequalities

the upper bound (UB), lower bound (LB), percentage gap (%), and run time in seconds (Time (s)) obtained with each of the two algorithms. B&B-1 was not able to find a feasible solution for instances (100/3/h/3) and (100/6/l/2), thus, the average indicated with * does not include those instances. The number of instances solved to optimality increases from 25 for B&B-1, to 32 out of 80 for B&B-2. The average optimality gap obtained by B&B-2 on each row is smaller than the average of B&B-1, with an average value of 4.9% against 11.8%. Algorithm B&B-2 outperforms B&B-1 in 45 instances (including the two where B&B-1 fails to find a feasible solution), ties in 30, and loses in only five. The reason why B&B-2 outperforms B&B-1 is related to the fact that B&B-2 includes the subset of the most effective inequalities only (see Section 4.2), while all inequalities (11)–(16) were all included in B&B-1, thus slowing down the solution process.

The performance of B&B-2 can be improved by the VMND algorithm, which embeds the local search in B&B-2. The results of Table 4 highlight that not only the VMND algorithm is capable of finding better solutions, but it also slightly improves the lower bound and the running time. The VMND algorithm solves 32 instances, the same number as B&B-2, but reduces the optimality gap on average from 4.9% to 3.4%. VMND beats B&B-2 in 33 instances, ties in 40, and loses in seven.

| Instance | B&B-1 | | | | B&B-2 | | | |
|---|---|---|---|---|---|---|---|---|
| | UB | LB | Gap (%) | Time (s) | UB | LB | Gap (%) | Time (s) |
| (25/3/h) | 3641.3 | 3640.9 | 0.0 | 229.2 | 3641.3 | 3641.3 | 0.0 | 107.6 |
| (50/3/h) | 4874.9 | 4739.2 | 2.7 | 5093.8 | 4841.6 | 4841.6 | 0.0 | 1605.8 |
| (75/3/h) | 11127.7 | 9976.6 | 10.4 | 7200.0 | 10574.3 | 10258.8 | 3.0 | 7200.0 |
| (100/3/h) | 14882.1* | 13081.4 | 11.9 | 7200.0 | 14480.4 | 13340.3 | 8.0 | 7200.0 |
| (25/6/h) | 5778.6 | 5778.2 | 0.0 | 172.6 | 5778.6 | 5778.6 | 0.0 | 39.6 |
| (50/6/h) | 7919.5 | 7171.4 | 9.4 | 7200.0 | 7909.1 | 7735.1 | 2.1 | 5603.4 |
| (75/6/h) | 8286.7 | 6823.7 | 17.6 | 7200.0 | 8074.0 | 7686.6 | 4.7 | 7200.0 |
| (100/6/h) | 11499.0 | 8001.4 | 29.8 | 7200.0 | 10121.1 | 9062.5 | 10.3 | 7200.2 |
| (25/3/l) | 2286.4 | 2286.2 | 0.0 | 43.8 | 2286.4 | 2286.4 | 0.0 | 25.4 |
| (50/3/l) | 3230.2 | 3177.5 | 1.6 | 5157.6 | 3223.2 | 3220.5 | 0.1 | 2367.0 |
| (75/3/l) | 4469.3 | 3852.5 | 13.7 | 7200.0 | 4330.5 | 4056.7 | 6.3 | 7200.0 |
| (100/3/l) | 5972.3 | 4642.8 | 21.9 | 7200.0 | 5525.2 | 4893.9 | 11.3 | 7200.2 |
| (25/6/l) | 4051.6 | 4051.2 | 0.0 | 1835.2 | 4051.6 | 4051.6 | 0.0 | 530.4 |
| (50/6/l) | 5637.3 | 4903.1 | 13.0 | 7200.0 | 5526.0 | 5341.2 | 3.3 | 6185.8 |
| (75/6/l) | 5877.1 | 4524.5 | 22.9 | 7200.0 | 5479.9 | 5051.6 | 7.7 | 7200.0 |
| (100/6/l) | 8483.8* | 5515.6 | 33.6 | 7200.0 | 7779.4 | 6017.3 | 21.4 | 7200.0 |
| Average | 6751.1 | 5760.4 | 11.8 | 5283.3 | 6476.4 | 6079.0 | 4.9 | 4629.1 |

*The algorithm was not able to solve instances (100/3/h/3) and (100/6/l/2).

**Table 3:** Comparison of the performance of exact methods

| Instance | B&B-2 | | | | VMND | | | |
|---|---|---|---|---|---|---|---|---|
| | UB | LB | Gap (%) | Time (s) | UB | LB | Gap (%) | Time (s) |
| (25/3/h) | 3641.3 | 3641.3 | 0.0 | 107.6 | 3641.3 | 3641.3 | 0.0 | 328.4 |
| (50/3/h) | 4841.6 | 4841.6 | 0.0 | 1605.8 | 4841.6 | 4841.6 | 0.0 | 1100.2 |
| (75/3/h) | 10574.3 | 10258.8 | 3.0 | 7200.0 | 10631.4 | 10249.0 | 3.5 | 7200.0 |
| (100/3/h) | 14480.4 | 13340.3 | 8.0 | 7200.0 | 13971.5 | 13348.5 | 4.5 | 7200.0 |
| (25/6/h) | 5778.6 | 5778.6 | 0.0 | 39.6 | 5778.6 | 5778.6 | 0.0 | 236.8 |
| (50/6/h) | 7909.1 | 7735.1 | 2.1 | 5603.4 | 7909.1 | 7728.8 | 2.2 | 5827.6 |
| (75/6/h) | 8074.0 | 7686.6 | 4.7 | 7200.0 | 8071.6 | 7692.2 | 4.6 | 7200.0 |
| (100/6/h) | 10121.1 | 9062.5 | 10.3 | 7200.2 | 9682.0 | 9058.5 | 6.4 | 7200.0 |
| (25/3/l) | 2286.4 | 2286.4 | 0.0 | 25.4 | 2286.4 | 2286.4 | 0.0 | 92.6 |
| (50/3/l) | 3223.2 | 3220.5 | 0.1 | 2367.0 | 3223.0 | 3223.0 | 0.0 | 1934.8 |
| (75/3/l) | 4330.5 | 4056.7 | 6.3 | 7200.0 | 4334.5 | 4052.4 | 6.4 | 7200.0 |
| (100/3/l) | 5525.2 | 4893.9 | 11.3 | 7200.2 | 5336.2 | 4900.5 | 8.2 | 7200.0 |
| (25/6/l) | 4051.6 | 4051.6 | 0.0 | 530.4 | 4051.6 | 4051.6 | 0.0 | 781.2 |
| (50/6/l) | 5526.0 | 5341.2 | 3.3 | 6185.8 | 5523.4 | 5380.5 | 2.5 | 6438.8 |
| (75/6/l) | 5479.9 | 5051.6 | 7.7 | 7200.0 | 5377.4 | 5061.4 | 5.8 | 7200.0 |
| (100/6/l) | 7779.4 | 6017.3 | 21.4 | 7200.0 | 6687.8 | 6013.6 | 10.0 | 7200.0 |
| Average | 6476.4 | 6079.0 | 4.9 | 4629.1 | 6334.2 | 6081.8 | 3.4 | 4646.3 |

**Table 4:** Comparison of the performance of exact methods

| Instance | B&B-1 | B&B-2 | VMND |
|---|---|---|---|
| (25/3/h) | 5 | 5 | 5 |
| (50/3/h) | 3 | 5 | 5 |
| (75/3/h) | 0 | 2 | 3 |
| (100/3/h) | 0 | 0 | 5 |
| (25/6/h) | 5 | 5 | 5 |
| (50/6/h) | 3 | 5 | 5 |
| (75/6/h) | 0 | 3 | 4 |
| (100/6/h) | 0 | 1 | 4 |
| (25/3/l) | 5 | 4 | 4 |
| (50/3/l) | 4 | 4 | 5 |
| (75/3/l) | 1 | 2 | 2 |
| (100/3/l) | 1 | 0 | 4 |
| (25/6/l) | 5 | 5 | 5 |
| (50/6/l) | 0 | 3 | 5 |
| (75/6/l) | 0 | 1 | 5 |
| (100/6/l) | 0 | 0 | 5 |
| Success rate (%) | 40.0 | 56.3 | 88.8 |

**Table 5:** Number of best UB found by algorithm

A summary that compares B&B-1, B&B-2 and VMND is provided in Table 5 where the number of best upper bounds found by each algorithm is shown. The table clearly shows that VMND outperforms the other two approaches when the size of the instances increases. When the size is small, the three algorithms performs almost the same, as the instances can be solved to optimality by all of them in the majority of the cases. In Table 6 the time required by B&B-2 and VMND to find their best upper bounds is shown in the second and third columns. The fourth column shows how long it took VMND to find an upper bound at least as good as the best solution found by B&B-2. It can be observed that VMND finds its best solution in about one third of the time it takes B&B-2 to reach its best solution, and that it takes VMND about one fifth of the time it takes B&B-2 to reach a similar solution.

In order to have an understanding of the structure of the tested instances, Table 7 shows the average percentage of each cost component in the best solutions found by VMND. As in other problems with the same structure, routing costs dominate the solution.

Finally, Tables 8 and 9 show the results obtained by VMND on instances with two and three vehicles, respectively. The average optimality gap is in each case equal to 4.4% (recall that it was 3.4% on instances with one vehicle). These values indicate that the algorithm performance is robust with respect to the increase of the number of vehicles.

# 5 Conclusion

In this paper we studied the vehicle routing problem with due dates where customers are associated with release and due dates and a distribution plan has to be built over a planning horizon with the aim of minimizing the sum of routing cost, inventory cost and the cost for postponing customer visits. A Variable MIP Neighborhood Descent algorithm (VMND) is presented where the idea is to embed a local search in an exact scheme and define search operators through variable fixing on MILPs. The algorithm improves 44 out of 90 best known solutions on benchmark instances where one vehicle is available. On instances with two and three vehicles, not previously tested, the average optimality

| Instance | B&B-2 | VMND | |
|---|---|---|---|
| | UB time (s) | UB time (s) | Time to beat B&B-2 (s) |
| (25/3/h) | 53.4 | 57.4 | 57.4 |
| (50/3/h) | 1344.2 | 563.6 | 563.6 |
| (75/3/h) | 3520.0 | 1532.2 | 1499.4 |
| (100/3/h) | 6519.8 | 2583.8 | 169.4 |
| (25/6/h) | 21.2 | 29.0 | 29.0 |
| (50/6/h) | 2082.0 | 203.2 | 203.2 |
| (75/6/h) | 3340.8 | 612.2 | 609.8 |
| (100/6/h) | 5089.6 | 3274.0 | 1342.8 |
| (25/3/l) | 6.8 | 10.4 | 10.4 |
| (50/3/l) | 1726.0 | 584.0 | 584.0 |
| (75/3/l) | 2758.0 | 1592.8 | 1585.2 |
| (100/3/l) | 6186.0 | 1964.2 | 94.2 |
| (25/6/l) | 90.2 | 18.4 | 18.4 |
| (50/6/l) | 2538.6 | 1032.0 | 1032.0 |
| (75/6/l) | 3463.0 | 557.8 | 505.2 |
| (100/6/l) | 6322.8 | 1568.2 | 1456.0 |
| Average | 2816.4 | 1011.5 | 610.0 |

**Table 6:** Comparison of the times to find the best upper bound of B&B-2 and VMND

| Instance | Routing (%) | Inventory (%) | Penalty (%) |
|---|---|---|---|
| (25/3/h) | 79.5 | 17.2 | 3.3 |
| (50/3/h) | 77.9 | 21.2 | 0.9 |
| (75/3/h) | 42.4 | 4.5 | 53.1 |
| (100/3/h) | 39.0 | 4.9 | 56.1 |
| (25/6/h) | 81.0 | 16.3 | 2.7 |
| (50/6/h) | 86.6 | 12.3 | 1.1 |
| (75/6/h) | 83.7 | 14.4 | 1.9 |
| (100/6/h) | 87.7 | 11.8 | 0.5 |
| (25/3/l) | 71.6 | 3.0 | 25.4 |
| (50/3/l) | 71.5 | 4.2 | 24.3 |
| (75/3/l) | 62.2 | 3.0 | 34.7 |
| (100/3/l) | 69.6 | 2.6 | 27.8 |
| (25/6/l) | 83.0 | 3.2 | 13.8 |
| (50/6/l) | 86.4 | 5.0 | 8.6 |
| (75/6/l) | 79.8 | 3.8 | 16.4 |
| (100/6/l) | 77.5 | 3.5 | 19.0 |
| Average | 73.7 | 8.2 | 18.1 |

**Table 7:** Composition of the costs for the best solution found with VMND

| Instance | UB | LB | GAP (%) | Time (s) | Routing (%) | Inventory (%) | Penalty (%) |
|---|---|---|---|---|---|---|---|
| (25/3/h) | 3634.7 | 3634.7 | 0.0 | 518.2 | 80.1 | 17.4 | 2.5 |
| (50/3/h) | 4841.6 | 4841.6 | 0.0 | 1362.6 | 77.9 | 21.2 | 0.9 |
| (75/3/h) | 6403.4 | 5922.7 | 7.4 | 7200.0 | 92.0 | 6.7 | 1.4 |
| (100/3/h) | 7542.7 | 6705.4 | 11.0 | 7200.0 | 94.4 | 5.6 | 0.0 |
| (25/6/h) | 5711.4 | 5711.4 | 0.0 | 212.4 | 81.4 | 15.8 | 2.7 |
| (50/6/h) | 7875.3 | 7696.9 | 2.2 | 5823.8 | 86.0 | 13.0 | 1.1 |
| (75/6/h) | 8080.1 | 7666.9 | 5.1 | 7200.0 | 83.8 | 14.4 | 1.9 |
| (100/6/h) | 9753.4 | 9041.6 | 7.3 | 7200.0 | 89.2 | 10.3 | 0.5 |
| (25/3/l) | 2286.4 | 2286.4 | 0.0 | 120.0 | 71.6 | 3.0 | 25.4 |
| (50/3/l) | 3189.6 | 3177.9 | 0.4 | 3811.4 | 75.6 | 5.5 | 18.9 |
| (75/3/l) | 4235.1 | 3894.6 | 8.0 | 7200.0 | 65.0 | 3.1 | 31.9 |
| (100/3/l) | 5254.6 | 4609.9 | 12.3 | 7200.0 | 70.5 | 2.6 | 26.9 |
| (25/6/l) | 3993.9 | 3970.7 | 0.6 | 2319.6 | 81.8 | 4.1 | 14.0 |
| (50/6/l) | 5501.7 | 5310.7 | 3.4 | 7200.0 | 86.0 | 5.6 | 8.4 |
| (75/6/l) | 5349.4 | 5010.8 | 6.2 | 7200.0 | 80.6 | 4.3 | 15.1 |
| (100/6/l) | 6461.9 | 5992.9 | 7.2 | 7200.0 | 79.9 | 4.9 | 15.2 |
| Average | 5632.2 | 5342.2 | 4.4 | 4935.5 | 81.0 | 8.6 | 10.4 |

**Table 8:** Results for the VMND algorithm for the two vehicle case

| Instance | UB | LB | GAP (%) | Time (s) | Routing (%) | Inventory (%) | Penalty (%) |
|---|---|---|---|---|---|---|---|
| (25/3/h) | 3634.7 | 3634.7 | 0.0 | 412.6 | 80.1 | 17.4 | 2.5 |
| (50/3/h) | 4841.6 | 4841.6 | 0.0 | 1247.8 | 77.9 | 21.2 | 0.9 |
| (75/3/h) | 6310.8 | 5932.4 | 5.9 | 7200.0 | 89.0 | 9.7 | 1.3 |
| (100/3/h) | 7479.7 | 6761.1 | 9.6 | 7200.0 | 93.4 | 6.6 | 0.0 |
| (25/6/h) | 5711.4 | 5711.4 | 0.0 | 159.4 | 81.4 | 15.8 | 2.7 |
| (50/6/h) | 7868.8 | 7720.6 | 1.8 | 5789.0 | 85.6 | 13.3 | 1.1 |
| (75/6/h) | 8070.5 | 7675.8 | 4.8 | 7200.0 | 83.5 | 14.6 | 1.9 |
| (100/6/h) | 9673.3 | 9037.3 | 6.5 | 7200.0 | 87.6 | 11.9 | 0.5 |
| (25/3/l) | 2286.4 | 2286.4 | 0.0 | 87.2 | 71.6 | 3.0 | 25.4 |
| (50/3/l) | 3190.2 | 3159.6 | 0.9 | 4079.8 | 76.5 | 4.8 | 18.7 |
| (75/3/l) | 4228.1 | 3886.5 | 8.0 | 7200.0 | 66.2 | 3.2 | 30.6 |
| (100/3/l) | 5234.8 | 4553.6 | 12.9 | 7200.0 | 72.6 | 4.6 | 22.6 |
| (25/6/l) | 3993.9 | 3989.6 | 0.1 | 2087.4 | 81.8 | 4.1 | 14.0 |
| (50/6/l) | 5501.7 | 5297.6 | 3.7 | 7200.0 | 86.0 | 5.6 | 8.4 |
| (75/6/l) | 5354.1 | 4999.7 | 6.5 | 7200.0 | 79.6 | 4.1 | 16.3 |
| (100/6/l) | 6582.8 | 5991.2 | 8.9 | 7200.0 | 78.1 | 4.2 | 17.7 |
| Average | 5622.7 | 5342.4 | 4.4 | 4916.5 | 80.7 | 9.0 | 10.3 |

**Table 9:** Results for the VMND algorithm for the three vehicle case.

is 4.4%. The results show that the local search is able to speed up the exact solution algorithm which then finds better solutions in a shorter computational time than the pure branch-and-bound algorithm. In addition, the results confirm that the VMND, proposed in the literature for a different problem, is easily adaptable to other problems and may be helpful in enhancing exact algorithms for combinatorial optimization problems.

# References

[1] 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.

[2] 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.

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

[4] C. Archetti, O. Jabali, and M. G. Speranza. Multi-period vehicle routing problem with due dates. *Computers & Operations Research*, 61(Supplement C):122–134, 2015.

[5] C. Archetti, N. Boland, and M. G. Speranza. A matheuristic for the multi-vehicle inventory routing problem. *INFORMS Journal on Computing*, 29:377–387, 2017.

[6] C. Archetti, E. Fernández, and D. L. Huerta-Muñoz. The flexible periodic vehicle routing problem. *Computers & Operations Research*, 85:58 – 70, 2017.

[7] G. Bertazzi, L. Paletta and M.G. Speranza. Minimizing the total cost in an integrated vendormanaged inventory system. *Journal of Heuristics*, 11(5–6):393–419, 2005.

[8] L. Bertazzi and M.G. Speranza. Inventory routing problems: an introduction. *EURO Journal on Transportation and Logistics*, 1(4):307–326, 2012.

[9] L. Bertazzi and M.G. Speranza. Inventory routing problems with multiple customers. *EURO Journal on Transportation and Logistics*, 2(3):255–275, 2013.

[10] 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.

[11] L. C. Coelho and G. Laporte. Improved solutions for inventory-routing problems through valid inequalities and input ordering. *International Journal of Production Economics*, 155(1):391–397, 2014.

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

[13] L. C. Coelho, J.-F. Cordeau, and G. Laporte. Consistency in multi-vehicle inventory-routing. *Transportation Research Part C: Emerging Technologies*, 24(1):270–287, 2012.

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

[15] G. Desaulniers, J. G. Rakke, and L. C. Coelho. A branch-price-and-cut algorithm for the inventory-routing problem. *Transportation Science*, 50(3):1060–1076, 2016.

[16] P. M. Francis, K. R. Smilowitz, and M. Tzur. The period vehicle routing problem and its extensions. In B.L. Golden, S. Raghavan, and E. A. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 73–102. Springer US, Boston, MA, 2008.

[17] S. Irnich, M. Schneider, and D. Vigo. Four variants of the vehicle routing problem. In P. Toth and D. Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications*, pages 241–260. MOS-SIAM Series on Optimization, Philadelphia, 2014.

[18] S. Irnich, P. Toth, and D. Vigo. The family of vehicle routing problems. In P. Toth and D. Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications*, pages 1–33. MOS-SIAM Series on Optimization, Philadelphia, 2014.

[19] H. Larrain, L. C. Coelho, and A. Cataldo. A variable MIP neighborhood descent algorithm for managing inventory and distribution of cash in automated teller machines. *Computers & Operations Research*, 85(Supplement C):22–31, 2017.