CIRRELT

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

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

# The Multi-Vehicle Traveling Purchaser Problem with Pairwise Incompatibility Constraints and Unitary Demands: A Branch-and-Price Approach

**Daniele Manerba**
**Michel Gendreau**
**Renata Mansini**

**October 2014**

**CIRRELT-2014-52**

**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

# The Multi-Vehicle Traveling Purchaser Problem with Pairwise Incompatibility Constraints and Unitary Demands: A Branch-and-Price Approach

**Daniele Manerba[1], Michel Gendreau[2], Renata Mansini[1]**

[1] Department of Information Engineering, University of Brescia, via Branze 38, 25123 Brescia, Italy

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Mathematics and Industrial Engineering, Polytechnique Montréal, P.O. Box 6079, Station Centre-Ville, Montréal, Canada H3C 3A7

**Abstract.** In this work we study a suppliers selection and routing problem where a fleet of homogeneous vehicles with a predefined capacity is available for procuring different products from different suppliers with the aim to minimize both the traveling and the purchasing costs. Decisions are further complicated by the presence of pairwise incompatibility constraints among products, implying the impossibility of loading two incompatible products on the same vehicle. The problem is known as the Multi-Vehicle Traveling Purchaser Problem with Pairwise Incompatibility Constraints. We study a variant in which products demand is unitary and propose a column generation approach based on a Dantzig-Wolfe reformulation of the problem, where each column represents a feasible vehicle route associated with a compatible purchasing plan. Two different procedures are introduced to solve the pricing problem, namely a labeling algorithm solving a Resource-Constrained Elementary Shortest Path Problem on an expanded graph, and a tailored branch-and-cut algorithm. Due to the integrality request on variables, we embed the column generation in a branch-and-bound framework and propose different branching rules, thus obtaining a branch-and-price procedure. Extensive tests, carried out on a large set of instances, show that our branch-and-price method performs well, improving on average, both in quality and in computational time, solutions obtained by a branch-and-cut approach existing in the literature that relies on a three-index connectivity constraints based formulation.

**Keywords**. Traveling purchaser problem, multi-vehicle, pairwise incompatibility constraints, column generation, branch-and-price.

**Acknowledgements.** This work was partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through its discovery grants program.



---

* Corresponding author: daniele.manerba@unibs.it

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2014

# 1 Introduction

The Travelling Purchaser Problem (TPP) is a single-vehicle routing problem that has been widely studied in the last decades. Consider a depot 0, a set $M$ of suppliers dispersed in a geographical area around the depot, and a set $K$ of products to purchase. A purchasing demand $d_k$, associated with each product $k \in K$, can be satisfied in a subset $M_k \subseteq M$ of suppliers at a non-negative price $p_{ik}$, potentially different for each supplier $i \in M_k$. For each product $k \in K$, a quantity $q_{ik}$ is offered by supplier $i \in M_k$, such that $\sum_{i \in M_k} q_{ik} \geq d_k$. Let $G := (V, A)$ be a complete directed graph where $V := \{0\} \cup M$ is the node set and $A := \{(i, j) : i \in V, j \in V, i \neq j\}$ is the arc set. A traveling cost $c_{ij}$ is associated with each arc $(i, j) \in A$. The TPP looks for a cycle in $G$ starting and ending to the depot, and visiting a subset of suppliers so to exactly satisfy products demand, while minimizing both traveling and purchasing costs. This problem variant is known as *restricted* TPP. The *unrestricted* TPP is instead an interesting special case where supplies are unlimited (i.e., $q_{ik} \geq d_k, k \in K, i \in M_k$), that is equivalent to consider $d_k = 1$, $k \in K$, and $q_{ik} = 1$, $k \in K$, $i \in M_k$. The combined optimization of routing decisions, suppliers selection and purchasing plan construction makes the TPP a very challenging problem, that fits well in several real-life procurement settings and in other contexts like job scheduling (Burstall, 1966). Hence, a lot of exact and heuristic approaches have been developed by researchers for the TPP and its variants (see, e.g., Voss, 1996, Laporte et al., 2003, and Mansini and Tocchella, 2009).

Recently, a multi-vehicle variant of the Travelling Purchaser Problem (MVTPP) has been introduced by Choi and Lee (2011) as a formulation for maximizing the reliability of components in a purchasing system with budget constraints. In the MVTPP, optimization has to be done over a fleet $F$ of homogeneous vehicles, ensuring that the quantity collected by each vehicle does not exceed a predefined capacity $Q$. Only few contributes can be found in the literature for this problem. Riera-Ledesma and Salazar-González (2012) study the MVTPP as a location routing problem in the context of school bus service, where suppliers correspond to bus stops and products to students to pick up. They present a branch-and-cut approach based on a single-commodity flow formulation enhanced by valid inequalities. The same authors propose a column generation approach for the problem (Riera-Ledesma and Salazar-González, 2013) taking into account different resource constraints for each route such as upper bounds on the distances traveled by the students, upper bounds on the number of visited bus stop and lower bounds on the number of students to pick up. Finally, Bianchessi et al. (2014) study the distance-constrained MVTPP ensuring that the distance traveled by each vehicle does not exceed a predefined upper bound.

More recently, in Manerba and Mansini (2014) the authors introduce a generalization of the MVTPP further complicated by the presence of pairwise incompatibilities among products imposing the impossibility of loading two incompatible products on the same vehicle.

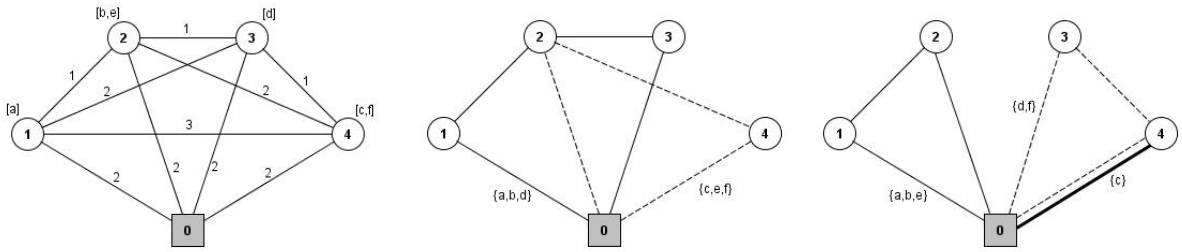
This problem is called Multi-Vehicle Traveling Purchaser Problem with Pairwise Incompatibility Constraints (MVTPP-PIC). The MVTPP-PIC aims at determining a tour for each vehicle $v \in F$ visiting a subset of suppliers so that the total product demands are satisfied at minimum purchasing and traveling cost, while guaranteeing all existing incompatibility restrictions among products and without exceeding vehicle capacity $Q$. While in general the use of more than one vehicle highly increases the complexity of a routing problem, in this case the problem complexity strongly depends also on the number of products taken into account and on the number of incompatibilities existing among them.

In the present paper, we study the MVTPP-PIC with unitary demands assuming $d_k = 1$, $k \in K$. Since the problem does not allow to purchase more than the required demand, the availability of a product is also assumed to be unitary in each supplier that sells it, i.e. $q_{ik} = 1$, $k \in K$, $i \in M_k$. Note that, in a multi-vehicle setting, this is not equivalent to the unrestricted case of the MVTPP-PIC with $d_k \geq 1$, due to the presence of a capacity constraint on vehicles. For modeling the incompatibility issues, let us define $B := \{(k, g) : k, g \in K\}$ as the set of pairs of products that cannot be transported on the same vehicle, simultaneously. We state here some definitions used throughout the paper. A product is said to be *incompatible* if it appears in at least one pair of the set $B$. On the contrary, a product is said to be *free* when it is compatible with all the other products and can be loaded freely on any vehicle. Finally, a product is said to be *totally incompatible* if it is in conflict with the entire set of incompatible products. Each totally incompatible product can be transported only with free products.

Similarly to the Split-Delivery Vehicle Routing Problems (SDVRP), the MVTPP-PIC allows multiple visits to the same supplier by different vehicles. However, while in SDVRP the split of deliveries might lead to savings in the traveling cost, this may not be the case for the MVTPP-PIC. Due to the incompatibilities among products, multiple visits of a supplier could be necessary, possibly impacting in a negative way on the cost. Example 1 shows the optimal solution of the MVTPP-PIC on two instances differing only for the set of incompatibilities. In the first case a multiple visit results to be convenient, whereas in the second case it is induced by the incompatibility constraints.

**Example 1** *Let us consider a fleet of vehicles with capacity* $Q = 3$ *placed at a depot* 0, *6 products* $\{a, b, c, d, e, f\}$, *and* 4 *suppliers* $\{1, 2, 3, 4\}$ *offering a subset of those products as shown in square brackets in Figure 1(a). The same figure shows the topology of the graph and the traveling costs associated with each pair of nodes. In Figures 1(b) and 1(c) are plotted the visiting tours of the vehicles in the MVTPP-PIC optimal solution when the set* $B$ *of incompatibilities is* $\{(a, c), (b, c), (c, d)\}$ *and* $\{(a, c), (b, c), (c, d), (c, f)\}$, *respectively. In the first case, the multiple visit of supplier* 2 *is convenient. In fact, if multiple visits were not allowed, one more vehicle would be necessary to satisfy the demand and the total cost would*

*rise to 13. In the second case, visiting twice supplier 4 becomes compulsory because of the incompatibility between products c and f (no feasible solution would exists if multiple visits were not allowed), thus impacting negatively on the cost.*




(a) Instance (b) Traveling cost: 12 (c) Traveling cost: 14

Figure 1: Convenient and necessary multiple visits in MVTPP-PIC solutions.

## 1.1 Motivations and contributions

Incompatibility restrictions can be found in several real-life problems involving the loading of vehicles. For example, in general markets procurement, foods and chemical products should be transported separately, and again in industry distribution, hazardous materials/liquids might dangerously react if loaded together on the same vehicle (Christofides et al., 1979). Frequently, in the literature, the incompatibility issue is tackled through the use of heterogeneous dedicated vehicles or through the introduction of separation devices on the vehicles (the latter problem, known as Multi-Compartment Vehicle Routing Problem, is surveyed in Iori and Martello, 2010). From a theoretical point of view, similar restrictions on items (sometimes called *exclusionary side constraints*) have been studied in the combinatorial optimization literature. Such constraints can be found, for instance, applied to the Transportation Problem (see Sun, 2002 and Goossens and Spieksma, 2009) and to the Bin Packing Problem (Gendreau et al., 2004).

The only existing contribution applying incompatibility constraints to a vehicle routing problem can be found in Manerba and Mansini (2014), where the MVTPP-PIC with non-unitary demands is studied. The authors first propose a matheuristic that relies on a decomposition of the original problem in four subproblems treated separately. Then, the best solution found by this heuristic is used as starting solution for a branch-and-cut method based on the separation of several families of valid inequalities. The overall exact approach is able to widely outperform, within 2 hours of computational time, the performance of the commercial MIP solver Cplex 12.3 tackling a compact three-index one-commodity flow formulation. However, as it frequently happens to branch-and-cut approaches, it presents some evident drawbacks:

- the three-index formulation used for the branch-and-cut naturally incorporates symmetry issues. Notwithstanding the introduction of symmetry breaking constraints, the decision tree of the solution procedure tends to explode;
- the method does not seem to scale very well. This is especially true for instances with a fleet size greater than 6-7 vehicles and for instances with a high density of incompatibility constraints.

The aim of the present work is to develop and test a new exact branch-and-price approach based on a set covering formulation of the problem, in order to efficiently overcome the drawbacks showed above. In particular, we propose a new formulation for the MVTPP-PIC that exploits the separability of the problem over the vehicles, thus solving symmetry issues. We will show that the new branch-and-price approach can efficiently optimize instances over a very large fleet of vehicles and with a high density of incompatibility among products. Finally, both the multi-vehicle optimization and the incompatibility constraints applied to the TPP are very recent streams of research, that deserve to be further analyzed.

The rest of the paper is structured as follows. In Section 2, we first present a mixed integer linear programming formulation of the problem based on connectivity constraints to exclude subtours, then we propose and analyze a set covering based reformulation obtained through a Dantzig-Wolfe decomposition technique. Section 3 is devoted to the description of the proposed branch-and-price approach. We present a simple initial heuristic, the column generation procedure along with two different exact algorithms for solving the pricing problem (i.e. a labeling algorithm on an expanded graph and a tailored branch-and-cut approach), and the branching rules to ensure integrality of the solutions. The computational results are discussed in Section 4, along with the presentation of the instances used. Finally, some conclusions are sketched in Section 5.

# 2 Mathematical formulations

In this section, we first propose a three-index connectivity constraints based formulation for the problem, then a set covering formulation based on a Dantzig-Wolfe decomposition.

## 2.1 Three-index formulation

Let $x_{ij}^v$, for each arc $(i,j) \in A$ and for each vehicle $v \in F$, be a binary variable taking value 1 if arc $(i,j)$ is crossed by vehicle $v$, and 0 otherwise. Let $y_i^v$, for each supplier $i \in M$ and for each vehicle $v \in F$, be a binary variable taking value 1 if the corresponding node $i$ is visited

by vehicle $v$, and 0 otherwise. Let $w_k^v$, for each product $k \in K$ and for each vehicle $v \in F$, be a binary variable taking value 1 if product $k$ is loaded on vehicle $v$, and 0 otherwise. Finally, let $z_{ik}^v$, for each product $k \in K$, for each supplier $i \in M_k$ and for each vehicle $v \in F$, be the quantity of product $k$ purchased by supplier $i$ and loaded on vehicle $v$. Moreover, given a set $S \subset V$, let $\delta^+(S)$ denote the set of arcs $(i,j)$ with $i \in S$ and $j \in V \setminus S$, and $\delta^-(S)$ denote the set of arcs $(i,j)$ with $j \in S$ and $i \in V \setminus S$. The MVTPP-PIC with unitary demands can be formulated as follows:

$$\min \sum_{v \in F} \sum_{(i,j) \in A} c_{ij} x_{ij}^v + \sum_{k \in K} \sum_{i \in M_k} \sum_{v \in F} p_{ik} z_{ik}^v \tag{1}$$

subject to

$$\sum_{i \in M_k} \sum_{v \in F} z_{ik}^v = 1 \quad k \in K \tag{2}$$

$$\sum_{(i,j) \in \delta^+(\{h\})} x_{ij}^v = \sum_{(i,j) \in \delta^-(\{h\})} x_{ij}^v = y_h^v \quad h \in M, v \in F \tag{3}$$

$$\sum_{(i,j) \in \delta^+(S)} x_{ij}^v \geq y_h^v \quad S \subseteq M, h \in S, v \in F \tag{4}$$

$$\sum_{k \in K} \sum_{i \in M_k} z_{ik}^v \leq Q \quad v \in F. \tag{5}$$

$$z_{ik}^v \leq y_i^v \quad k \in K, i \in M_k, v \in F \tag{6}$$

$$w_k^v + w_g^v \leq 1 \quad (k,g) \in B, v \in F \tag{7}$$

$$\sum_{i \in M_k} z_{ik}^v = w_k^v \quad k \in K, v \in F \tag{8}$$

$$x_{ij}^v \in \{0,1\} \quad (i,j) \in A, v \in F \tag{9}$$

$$y_i^v \in \{0,1\} \quad i \in M, v \in F \tag{10}$$

$$w_k^v \in \{0,1\} \quad k \in K, v \in F \tag{11}$$

$$z_{ik}^v \geq 0 \quad k \in K, i \in M_k, v \in F \tag{12}$$

Objective function (1) establishes the minimization of traveling and purchasing costs. Constraints (2) ensure that each product demand must be satisfied. In constraints (3) we impose that if vehicle $v$ visits a supplier $h$ (i.e. $y_h^v = 1$), it also has to leave it, while connectivity inequalities (4) allow to eliminate subtours. Constraints (5) impose that the quantity loaded on each vehicle does not exceed its capacity. Constraints (6) avoid a vehicle $v$ to load a product $k$ from a supplier $i$ not included in its visiting tour. Constraints (7) and (8) model the pairwise incompatibility among products. The first group of constraints states that for each pair of incompatible products $(k,g) \in B$ a vehicle $v$ can load at most one of the two.

The second set of constraints ensures that if $w_k^v = 0$, vehicle $v$ cannot load product $k$ from any suppliers of its tour, whereas if $w_k^v = 1$ the vehicle must purchase exactly one unit of the product from a supplier that sells it. Finally, (9)–(12) are variables non-negativity and integrality conditions. Note that $w$-variables can be easily eliminated from the model using equalities (8), but we keep them to clearly state the assignment of a product to a vehicle.

## 2.2 Dantzig-Wolfe decomposition

We now describe a reformulation of the model (1)–(12) obtained through the use of the Dantzig-Wolfe decomposition technique for linear programs (see Dantzig and Wolfe, 1960). Let $D = \{(x, y, w, z) : (x, y, w, z)$ satisfies (3)–(12)$\}$ be the feasible domain of the subproblem identified by constraints (3)–(12). This domain is decomposable over the vehicles in identical feasible domains $D^v = \{(x^v, y^v, w^v, z^v) : (x^v, y^v, w^v, z^v)$ satisfies (3)–(12) for $v \in F\}$. A solution $(x^v, y^v, w^v, z^v) \in D^v$ corresponds to a *feasible route* when it is associated with a *feasible purchasing plan* satisfying vehicle capacity and incompatibility restrictions. Given this subproblem domain, the Dantzig-Wolfe decomposition leads to the master program described in the following (a similar decomposition approach has been proposed in Jin et al., 2008 for solving the SDVRP, and in Desaulniers, 2010 for the SDVRP with Time Windows).

Let us define $R$ as the set of all feasible routes. A feasible route is basically a set of arcs representing a tour in $G$ starting and ending to the depot 0, and visiting a subset of suppliers that guarantees the accomplishment of a feasible purchasing plan. A feasible purchasing plan associated with a route represents the set of decisions about which product to purchase from which supplier belonging to the route, ensuring that: a) a product is purchased from a supplier only if it is available there; b) incompatible products cannot be part of the same purchasing plan; c) the total quantity of products purchased does not exceed the vehicle capacity. That said, a feasible solution for the MVTPP-PIC can be viewed as a collection of $O(|F|)$ routes in $R$ such that the demand for each product is satisfied. For each feasible route $r \in R$, we also define:

- $\theta^r$: an integer variable equal to the number of times route $r$ is assigned to a vehicle;
- $\delta_{ik}^r$: a coefficient equal to 1 if product $k$ is purchased from supplier $i$ in the route $r$;
- $a_i^r$: a coefficient equal to 1 if node $i$ is visited in the route $r$, and 0 otherwise;
- $b_{ij}^r$: a coefficient equal to 1 if the arc $(i, j)$ is traversed in the route $r$, and 0 otherwise;
- $c^r = \sum_{(i,j)\in r} c_{ij}$: the total traveling cost associated with the route $r$;
- $p^r = \sum_{k\in K} \sum_{i\in M_k} p_{ik}\delta_{ik}^r$: the total purchasing cost associated with the route $r$.

Finally, let $X_{ij}$ be an integer variable representing the number of times arc $(i,j) \in A$ is traversed, and let $Y_i$ be an integer variable representing the number of times node $i \in V$ is visited. With this notation, the MVTPP-PIC can be reformulated as follows:

$$\min \sum_{r \in R} (c^r + p^r)\theta^r \tag{13}$$

subject to

$$\sum_{r \in R} \theta^r \leq |F| \tag{14}$$

$$\sum_{i \in M_k} \sum_{r \in R} \delta^r_{ik} \theta^r \geq 1 \quad k \in K \tag{15}$$

$$\theta^r \in \mathbb{N} \quad r \in R \tag{16}$$

$$\sum_{r \in R} a^r_i \theta^r = Y_i \quad i \in V \tag{17}$$

$$\sum_{r \in R} b^r_{ij} \theta^r = X_{ij} \quad (i,j) \in A \tag{18}$$

$$Y_i \in \mathbb{N} \quad i \in V \tag{19}$$

$$X_{ij} \in \mathbb{N} \quad (i,j) \in A \tag{20}$$

Objective function (13) minimizes the total cost of the selected routes. Inequality (14) limits the number of selected routes to the fleet size, while constraints (15) ensure that, for each product, its unitary demand is satisfied. Equations (17) and (18) express $Y_i$ and $X_{ij}$ variables in terms of the $\theta^r$ ones. Finally, in (16), (19) and (20) the integrality of variables is defined. Constraints (17)–(20) are not strictly necessary for the model. Anyway, these relations are stated here since they are useful to understand the branching rules proposed in Section 3.3. From now on, we denote the linear relaxation of (13)–(16) as *master problem* (MP).

Since the set $R$ contains a huge number of feasible routes, it is impossible to solve explicitly the model by generating all its variables. Hence, we resort to a branch-and-price solution approach where, at each node of the tree, the MP is solved by using a column generation procedure. Note that, differently from formulation (1)–(12), here it is not explicitly required to purchase exactly a unit for each product; however, this relaxation is valid since it is never optimal to buy more than one unit for any product. The rationale behind the choice of a set-covering formulation is that an equality in constraints (15) would lead to free dual variables, which typically slow down the convergence of a column generation algorithm. Finally, variables $\theta$ are not defined to be binary to avoid $|R|$ constraints of the type $\theta^r \leq 1$ in the MP, even if it is clear that any solution containing a route $r$ with $\theta^r > 1$ could not be optimal.

# 3 A branch-and-price solution approach

This section is devoted to describe all the components of the branch-and-price solution algorithm based on the set-covering formulation (13)–(20). Branch-and-price represents a standard approach for solving huge integer programs (see Barnhart et al., 1998) and it has been applied effectively to many vehicle routing and scheduling problems (see, e.g., Gendreau et al., 2006 and Desaulniers et al., 2002). This method combines a branch-and-bound framework with the use of a column generation technique for solving the linear relaxations at each node of the tree. In our implementation, the overall procedure is initialized by a simple heuristic, to find a feasible solution quickly.

## 3.1 Heuristic upper bound

In every enumerative methods based on bounding, it is useful to start the procedure with a feasible solution (an upper bound in this case) that permits to prune a higher number of nodes in the branch-and-bound tree. In order to better understand the behavior of our branch-and-price, we decide to implement just a basic initial heuristic consisting in a simplified version of the *four-step heuristic* proposed in Manerba and Mansini (2014). That heuristic relies on a decision tree, searched through a Beam Search procedure, where each node corresponds to a different subset of suppliers. In this work, that decision tree is essentially reduced to a unique node corresponding to the set of all the suppliers.

First, we find a feasible purchasing plan by optimally solving the following Assignment Problem, where $\lambda_{ik} := \sum_{v \in F} z_{ik}^v$ are continuous variables representing the quantity of product $k$ purchased in supplier $i$:

$$\min \sum_{k \in K} \sum_{i \in M_k} p_{ik} \lambda_{ik} \tag{21}$$

subject to

$$\sum_{i \in M_k} \lambda_{ik} = 1 \quad k \in K \tag{22}$$

$$\lambda_{ik} \geq 0 \quad k \in K, i \in M_k \tag{23}$$

Given the optimal solution of (21)–(23), we then eliminate all non-visited suppliers.

Since problem (21)–(23) ignores the vehicles, a greedy heuristic is then used to find a feasible assignment of products to vehicles and, consequently, of vehicles to suppliers. The method chooses randomly an incompatible product, and assign it to a vehicle. If a product can not be assigned to a vehicle (because it would exceed its capacity or because it is incompatible with some already loaded products) another vehicle is chosen, and so on. This procedure is repeated for each incompatible product. Then, for each supplier, the remaining free products fill up first the vehicles that have been already associated with that supplier.

Finally, once decided which suppliers have to be visited by each vehicle, we find a feasible route by solving a Traveling Salesman Problem (TSP). This is done heuristically through the Lin-Kernighan-Helsgaun algorithm, which is a recent variant of one of the most successful methods for generating near-optimal solutions for the TSP (Helsgaun, 2000).

## 3.2 Column generation

Column generation is a state-of-the-art method for solving linear programs containing a huge number of variables. In this method, columns (variables) are iteratively found by a so-called *pricing problem* (or *subproblem*), and added to the master problem. The pricing problem aims at computing columns with negative reduced cost with respect to a dual solution of the restricted master problem (RMP), i.e. the linear relaxation of the master problem limited to the current subset of variables. When no new columns with negative reduced cost can be found, the optimal solution of the RMP is also optimal for the MP.

In our case, given $\pi$ the non-positive dual variable associated with constraint (14), and $\mu_k$ the non-negative dual variable associated with a constraint (15) for product $k$, the pricing problem is as follows:

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{i\in M}\sum_{k\in K}(p_{ik} - \mu_k)z_{ik} - \pi \tag{24}$$

subject to

$$\sum_{(i,j)\in\delta^+(\{h\})} x_{ij} = \sum_{(i,j)\in\delta^-(\{h\})} x_{ij} = y_h \quad h \in M \tag{25}$$

$$\sum_{(i,j)\in\delta^+(S)} x_{ij} \geq y_h \quad S \subseteq M, h \in S \tag{26}$$

$$z_{ik} \leq y_i \quad k \in K, i \in M_k \tag{27}$$

$$\sum_{k\in K}\sum_{i\in M_k} z_{ik} \leq Q \tag{28}$$

$$w_k + w_g \leq 1 \quad (k,g) \in B \tag{29}$$

$$\sum_{i\in M_k} z_{ik} = w_k \quad k \in K \tag{30}$$

$$x_{ij} \in \{0,1\} \quad (i,j) \in A \tag{31}$$

$$y_i \in \{0,1\} \quad i \in M \tag{32}$$

$$w_k \in \{0,1\} \quad k \in K \tag{33}$$

$$z_{ik} \in \{0,1\} \quad k \in K, i \in M_k \tag{34}$$

Variables $x, y, w$ and $z$ have the same meaning as variables in formulation (1)–(12), but for a given vehicle $v \in F$. The problem (24)–(34) is a Traveling Purchaser Problem with Pairwise Incompatibility Constraints but without the demand constraints. In this problem, the purchasing of each product is not forced by a specific demand requirement (as in common TPP), but by the minimization of a purchasing cost where product (reduced) prizes $p'_{ik} := p_{ik} - \mu_k$ can be negative.

We propose two exact methods for solving the pricing problem, i.e. the labeling algorithm and the branch-and-cut approach described in Sections 3.2.1 and 3.2.2, respectively. It is worth noticing that, at each iteration of the column generation procedure, a lower bound on the optimal value of the MP ($\underline{z}_{MP}$) can be computed as:

$$\underline{z}_{MP} = z^*_{RMP} + \tilde{c}^*|F| \tag{35}$$

where $z^*_{RMP}$ corresponds to the optimal value of the current RMP and $\tilde{c}^*$ is the reduced cost value of the optimal column found by the relative pricing problem.

### 3.2.1 A labeling algorithm for the pricing problem

The pricing problem can be reformulated as an Elementary Shortest Path Problem with Resource Constraints (ESPPRC) on an expanded graph $G' = (A', V')$ defined as follows. The node set $V'$ includes the depot 0 and the nodes representing all the possible pairs $(k, i)$ where $k \in K$ and $i \in M_k$. The arc set $A'$ corresponds to the Cartesian product $V' \times V'$. For each arc $(v_1, v_2) \in A'$ we define a cost $\tilde{c}_{v_1,v_2}$ equal to:

$$\tilde{c}_{v_1,v_2} := \begin{cases} c_{0j} + p_{jk} - \mu_k & \text{if } v_1 = 0 \text{ and } v_2 = (k, j), \\ c_{ij} + p_{jg} - \mu_g & \text{if } v_1 = (k, i) \text{ and } v_2 = (g, j), \\ p_{ig} - \mu_g & \text{if } v_1 = (k, i) \text{ and } v_2 = (g, i), \\ c_{i0} & \text{if } v_1 = (k, i) \text{ and } v_2 = 0, \\ \infty & \text{otherwise.} \end{cases} \tag{36}$$

Even if solving an ESPPRC on $G'$ is an $\mathcal{NP}$-hard problem, the aim of this formulation is to take advantage from the effective labeling algorithms proposed in the literature (see e.g. Irnich and Desaulniers, 2005). In such labeling algorithms, any feasible partial path starting from the depot is represented by a label associated with the arriving node. Generally, labels include useful information about the partial path, such as its current (reduced) cost, the visited nodes, and the current value of its resources. Starting from an initial void label associated with the depot, labels are extended in all possible directions through arcs in the graph (and their components are updated accordingly), until they return to the depot. It is possible to extend a label only if the extension satisfies all the resource constraints. Finally, dominance procedures can be used to eliminate labels and speed up the solution. An exact

solution of the just explained algorithm returns a set of Pareto-optimal columns with respect to all the considered resources.

In our implementation, each feasible partial path from 0 to a node $v = (k, i) \in V'$ is represented by a label $E := (C, L, (t_k)_{k\in K}, v)$ where $C$ is the reduced cost of the partial path, $L$ represents the vehicle load, and $(t_k)_{k\in K}$ is a $\{0, 1\}$-vector indicating for each product $k$ if it is loaded on the vehicle. A partial path is feasible if the vehicle load does not exceed the capacity, i.e. $L \leq Q$, and if the vehicle does not transport incompatible products, i.e if for any $k, g \in K$ such that $t_k = 1$ and $t_g = 1$ it holds that $(k, g) \notin B$. Thus, a label can be extended only if the new label is associated with a feasible path. Extending a label $(C^{v_1}, L^{v_1}, (t_k^{v_1})_{k\in K}, v_1)$ along an arc $(v_1, v_2) \in A'$ to a node $v_2 = (g, j)$ produces a new label $(C^{v_2}, L^{v_2}, (t_k^{v_2})_{k\in K}, v_2)$ where $C^{v_2} = C^{v_1} + \tilde{c}_{v_1,v_2}$, $L^{v_2} = L^{v_1} + 1$, $t_g^{v_2} = 1$, and $t_k^{v_2} = t_k^{v_1}$ for $k \in K, k \neq g$. If instead the arriving node is the depot ($v_2 = 0$), the extension simply produces a new label $(C^{v_2}, L^{v_1}, (t_k^{v_1})_{k\in K}, v_2)$ where $C^{v_2} := C^{v_1} + \tilde{c}_{v_1,v_2}$.

Moreover, we decide to allow extensions of labels from a node $v_1 = (k, i)$ to a node $v_2 = (g, i)$ only if $g > k$. This restriction prevents the possibility of generating equivalent paths that differ only for the order in which products are purchased in a market, thus solving the symmetries yielded by using the expanded graph $G'$. Note also that it is never convenient to extend a label to a node $v_2 = (g, j)$ if $t_g^{v_1} = 1$, i.e. if product $g$ has been already loaded on the vehicle.

A dominance rule can be applied in order to eliminate partial paths that can not be extended to Pareto-optimal ones. Given two labels $E^1 = (C^1, L^1, (t_k^1)_{k\in K}, v)$ and $E^2 = (C^2, L^2, (t_k^2)_{k\in K}, v)$ associated with the same node $v$, $E^1$ is said to dominate $E^2$ if it collects the same products (independently from the visited markets) and it has a better reduced cost, i.e. if $t_k^1 = t_k^2, k \in K$ and $C^1 \leq C^2$.

### 3.2.2 A branch-and-cut algorithm for the pricing problem

In this section, we propose an alternative solution method to the dynamic programming based approach proposed above. Since several polyhedral aspects of (25)–(34) have been studied before, we are able to develop an efficient branch-and-cut algorithm by using some families of valid inequalities.

A consolidated approach in the literature consists in solving the model (25)–(34) without the connectivity constraints (26), that are exponential in the size of $|M|$, and in adding dynamically only those of them that better cutoff the current LP relaxation optimal solution $(x^*, y^*, w^*, z^*)$. An exact separation algorithm for connectivity constraints can be executed in polynomial time by solving max-flow/min-cut problems on a capacitated graph $G = (V, A)$ where for each arc $(i, j) \in A$ a capacity $x_{ij}^*$ is considered. More precisely, given a supplier $h$ such that $y_h^* \neq 0$, the most violated constraint (26) corresponds to the partition $(S, V \setminus S)$

associated with a minimum-capacity cut in $G$ separating node 0 from $h$, and such that $h \in S$.

We also consider valid inequalities corresponding to a particular type of connectivity constraints involving $z$-variables (the so-called zSEC inequalities introduced for the TPP by Laporte et al., 2003):

$$\sum_{(i,j)\in\delta^-(S)} x_{ij} + \sum_{h\in M_k\setminus S} z_{hk} \geq 1 \quad k \in K, S \subseteq M_k. \tag{37}$$

Each inequality (37) imposes that at least one arc must enter a subset $S \subseteq M_k$ whenever any product offered in $S$ is not purchased in at least one supplier of $M_k \setminus S$. Similarly to the previous separation procedure, given a product $k \in K$, we can formulate a max-flow/min-cut problem on a capacitated graph $G_k := (V_k, A_k)$ where $V_k := V \cup \{t\}$, $t$ is a dummy supplier, and $A_k = A \cup \{(h,t) : h \in M_k\}$. In this graph, the capacity of each arc $(i,j) \in A$ is equal to $x^*_{ij}$, whereas the capacity of each new arc $(h,t)$ is equal to $z^*_{hk}$. The most violated valid inequalities (37) corresponds to consider the set $S := S_t \setminus \{t\}$, where $(S_t, V_k \setminus S_t)$ is the node partition associated with a minimum-capacity cut in $G_k$ separating 0 and $t$, with $t \in S_t$. The cut is effective only if the minimum-capacity cut value is less than 1.

Moreover, the model (25)–(34) can be enriched with the direct addition of the following classes of simple valid inequalities:

$$z_{ik} \leq w_k \quad k \in K, i \in M_k, \tag{38}$$

$$\sum_{i\in M_k} y_i \geq w_k \quad k \in K, \tag{39}$$

$$y_i \leq \sum_{j\in M} x_{0j} \quad i \in M, \tag{40}$$

$$x_{ij} \leq y_j \quad i \in V, j \in M. \tag{41}$$

Inequalities (38) state that if a product $k$ is purchased at a supplier $i$ ($z_{ik} = 1$) then the product $k$ has to be loaded ($w_k = 1$). Inequalities (39) establish that if a product $k$ is loaded, at least one of the suppliers offering such a product has to be visited. Valid inequalities (40) simply state that if a supplier $i$ is visited, then at least an arc must leave the depot, and, through inequalities (25), that one arc has to enter and one has to leave the depot. The three previous cuts are inherited from those proposed in Manerba and Mansini (2014) for the MVTPP-PIC with non unitary demands and restricted availabilities. Finally, inequalities (41) are logical constraints ensuring that if the arc $(i,j)$ is used, then the supplier at the end of the arc has to be visited (Fischetti et al., 1998). These constraints make the integrality requirement on the $y$-variables redundant.

### 3.2.3 Stabilization

Column generation approaches generally suffer from slow convergence of the objective function. This is especially true when multiple dual solutions can be associated with each primal solution. A wrong choice of the dual variables values used in the subproblem could lead indeed to an inappropriate estimation of the marginal costs associated with columns. In order to overcome this drawback, a lot of different methods have been proposed in the literature (*stabilization* methods). Differently from those that try to limit the fluctuation of the dual variables values from an iteration to another (see e.g. du Merle et al., 1999), we decided to implement the interior point stabilization (IPS) method proposed by Rousseau et al. (2007). Briefly, the main idea is to solve, at each iteration of the column generation, a certain number of different slightly perturbed master problems, to collect all the optimal dual solutions corresponding (generally) to extreme points of the dual polyhedron, and finally to return a dual solution corresponding to an interior point of such a polyhedron. This point is calculated as the convex combination of all the collected extreme points.

The IPS method is particularly indicated when the MP needs very few CPU time to be solved, as in our case. On other hand, a better estimation of the marginal costs generally leads to subproblems more complex to solve. Hence, the method results efficient especially when is associated with heuristic or really optimized solution algorithms for the subproblem.

### 3.2.4 Accelerating techniques

Several techniques, described in the following, have been implemented in order to speed up the entire column generation phase.

***Initialization:*** We decide to initialize the column generation phase with columns corresponding to the routes originated from the initial heuristic (explained in Section 3.1). Hence, $O(|F|)$ columns are initially added. Moreover, in order to ensure the feasibility of the MP, even with respect to the branching constraints, we add $|K|$ artificial slack variables, one in each inequality (15). These variables are highly penalized in the objective function.

***Heuristic labeling algorithm:*** Since labeling algorithms are really time consuming, a common practice is to stop the generation of columns prematurely. This can be done apart in the very last iteration of the column generation, where we need to prove that no more negative reduced cost columns exist. We implement two different stopping rules based on the number of labels returned to the depot. More precisely, the algorithm stops when a number $negLabel_{LIM}$ of negative reduced cost labels has returned to the depot 0 (that corresponds to bound the number of columns added to the master problem for each iteration), or when

the total number of labels returned to the depot reaches $allLabel_{LIM}$. In order to make these stopping rules more efficient, they are checked every time a new label arrives in 0. Moreover, the first extension for each label associated with a node $v_1$ is along the arc $(v_1, 0)$.

***Returning multiple columns through the branch-and-cut algorithm:*** The branch-and-cut proposed in Section 3.2.2 aims at finding the optimal solution corresponding to a column with the minimum reduced cost. However, we store all the feasible integer solutions found during the branch-and-cut tree and return them as columns to add to the MP.

***Switching the subproblem solution methods:*** Actually, the heuristically stopped labeling algorithm and the branch-and-cut method proposed for solving the subproblem have quite complementary features. While the first method returns a lot of columns, most of which of bad quality, the second method returns very few columns but always including the best one. This suggests that the two methods can be used alternatively to solve the subproblems, taking advantage from the good features of each one. The combined use of the two algorithms might take different configurations. After several preliminary tests, we decide to implement the following scheme. At each node of the branch-and-price tree the subproblem is initially solved through the labeling algorithm, always checking for the stopping rules. If $negLabel_{LIM}$ is reached, the labeling algorithm stops and the MP (enriched by the columns found) is solved again, but the solution method for the subproblem does not change. Instead, it switches to the branch-and-cut as soon as the $allLabel_{LIM}$ is reached, i.e. when the labeling algorithm seems no more effective in finding negative reduced cost columns.

***Restricted master heuristic:*** We implement a two-phase procedure in order to get feasible solutions quickly without waiting the end of the entire process. This is useful also for reducing the number of visited nodes in the branch-and-price tree. Both the two phases are based on the solution of the RMP. First, during the column generation procedure, it might happen that the RMP yields an integer (non-optimal) solution but that such a solution results infeasible for the MVTPP-PIC showing a purchase greater than 1 for some products. In fact, the relaxation used in the master problem, i.e. the inequalities (15), is equivalent to the MVTPP-PIC only if optimality has been reached. However, given an integer solution, we can easily check if the purchase for each product is unitary, i.e. if $\sum_{i \in M_k} \sum_{r \in R} \delta_{ik}^r \theta^r = 1$, $k \in K$. In case of positive answer, we can update the best incumbent solution and the overall upper bound. In the negative case, the second phase is applied. This consists in optimally solving, through the use of a MIP solver, the integer model (13)–(16) restricted to the columns currently generated, and where the inequalities (15) are changed in equalities. If the optimal solution has a better value than the incumbent one, the upper bound is updated.

### 3.3 Branching

When the optimal solution of the MP results to be fractional, we need to implement a branching scheme to ensure integrality of variables. This rises some difficulties, because the branching rules need to be compatible with the column generation phase.

It is widely accepted that branching on variables $\theta$ representing routes in the master problem is inefficient, leading to strong imbalance in the branch-and-bound tree. Hence, we prefer to branch on the more representative variables derived from the connectivity constraints based formulation. Some very efficient branching schemes have been developed for branch-and-price approaches applied to similar vehicle routing problems where each supplier has to be visited at most by one vehicle. Unluckily, in our problem suppliers may be visited more than once, hence those schemes are not enough to ensure integrality. Following Gendreau et al. (2006), we propose a hierarchical branching scheme composed by three different rules. These branching decisions are explained in the order we apply them.

In the first branching rule, if any node is visited a fractional number $\alpha$ of times (i.e., if $Y_i$ is fractional), we choose the node $i$ for which the fractional part of the corresponding $Y_i$ variable's value is closest to 0.5, and we impose $\sum_{r\in R} a_i^r \theta^r \leq \lfloor \alpha \rfloor$ and $\sum_{r\in R} a_i^r \theta^r \geq \lceil \alpha \rceil$, respectively. Note that branching on the depot is equivalent to branching on the number of vehicles used in the solution.

When all $Y$ variables are integer, we check if any arc is traversed a fractional number $\beta$ of times (i.e., if any $X_{ij}$ is fractional). In this second branching rule, we select the arc $(i, j)$ with a fractional part of the corresponding $X_{ij}$ variable closest to 0.5 and we impose $\sum_{r\in R} b_{ij}^r \theta^r \leq \lfloor \beta \rfloor$ and $\sum_{r\in R} b_{ij}^r \theta^r \geq \lceil \beta \rceil$ for the two generated subproblems, respectively.

When all $Y$ and $X$ variables are integer, we adopt a third level of branching applying the so-called *Flow-Splitting Method* proposed in Feillet et al. (2005). Briefly, the idea is to branch on a subset of $\theta$ variables generating a fractional flow for the current solution. In vehicle routing problems this set can be generally identified by clustering variables that begin with the same sequence of visited customers. In our case, the clustering is based on the sequence of visits in an expanded graph where nodes are of the type $(k, i), k \in K, i \in M_k$, as in the graph $G'$ described in Section 3.2.1. Note that this third-level branching rules is not expected to be often used in practice, however it is necessary to ensure the $\theta$-variables integrality.

## 4 Computational experiments

This section presents the setting and the results of the computational experiments we run. The branch-and-price has been implemented in C++, makes use of Cplex 12.5 solver for the solution of linear programs, and the branch-and-bound tree is searched through a *best-first*

strategy. Cplex 12.5 is also used as a general branch-and-cut framework (for the solution of the subproblems), where cuts are embedded via Concert Technology callbacks. Concerning the IPS method, we generate (at each iteration) 20 random vertices of the optimal master problem's dual polyhedron through the simplex method. Preliminary tests confirmed that this number of points is large enough to produce a valuable dual variables stabilization effect, without damaging the total computational time.

## 4.1 Instances

No specific benchmark instances exist for the MVTPP-PIC with unitary demands. In Manerba and Mansini (2014) the authors introduced a set of instances for the more general case in which demands are not unitary. It is clear that a simple modification of those instances would lead to very meaningless relations between the demand of products and the capacity of the vehicles. We decide to create new instances adapting the method used in the just mentioned paper, as follows.

The suppliers and the depot integer coordinates are randomly generated in a $[0, 1000] \times [0, 1000]$ square according to a uniform distribution, and routing costs are computed by rounding the Euclidean distances through `EUC_2D` function from `TSPLIB`. Each product $k$ is randomly associated with $|M_k|$ selected suppliers, where $|M_k|$ is randomly generated in the interval $[1, |M|]$. For each product $k$ and each supplier $i$, prices $p_{ik}$ are selected in interval $[1, 200]$ according to a discrete uniform distribution. To generate the incompatibility set $B$, we first fix the percentage of free products with respect to the total number of products, i.e. $f\% = 100|K_{free}|/|K|$ where $K_{free}$ is the set of all the free products. Then, for each product $k \notin K_{free}$, we randomly choose the number and the set of products with which there is an incompatibility and we construct the incompatibility pairs. Parameter $f\%$ allows to better control the structure of the instance.

Finally, for each instance, we need to determine the number $|F|$ of available vehicles in order to guarantee that at least a feasible assignment of products to the vehicles exists. In Manerba and Mansini (2014) the authors propose an $\mathcal{NP}$-hard procedure (including the exact solution of a Minimum Vertex Coloring Problem) just to calculate the minimum feasible size of the fleet. The rationale is that, in the case of their branch-and-cut solution algorithm, the dimension of the model strongly depends on the number of vehicles. Since a strict evaluation of the fleet size is often impractical in real settings, we prefer to simply compute an upper bound on the optimal fleet size by using the following formula:

$$|F| := |K \setminus K_{free}| + max\left(0, \left\lceil \frac{|K_{free}| - |K \setminus K_{free}| * (Q-1)}{Q} \right\rceil\right). \tag{42}$$

Note that, if no incompatibility exists among products, expression (42) reduces to the common upper bound value used for computing the fleet size in vehicle routing problems.

## 4.2 Preliminary results

We run a set of preliminary tests to compare the performance of our branch-and-price approach with respect to the branch-and-cut method proposed in Manerba and Mansini (2014). The latter method has been used as a black-box, without adapting any procedure or the effectiveness of valid inequalities to the special case with unitary demands. We consider instances with a number of nodes (the suppliers plus the depot) $|V| = \{10, 25, 40\}$, and a number of products $|K| = \{10, 20, 30\}$. We also consider three percentage values of free products $f\%$, equal to 80%, 50% and 30%. This means that the most constrained instances have 70% of cross-incompatibilities between products. Moreover, we use three different values for the vehicle capacity $Q$, i.e. $\{6, 10, 15\}$. Generating an instance for each combination of $\{|V|, |K|, f\%, Q\}$ leads to a set of 81 instances. In these preliminary tests, we set the labeling algorithm's parameters $negLabel_{LIM}$ and $allLabel_{LIM}$ to 50 and 500, respectively. Moreover, for each solution method we set a threshold time of 7200 second (i.e., 2 hours).

Table 1 shows the comparison between the performance obtained, on the entire set of 81 instances, by our branch-and-price (`B&P`) and those obtained by the branch-and-cut (`MM-B&C`) method proposed in Manerba and Mansini (2014). Each table entry refers to the average result out of 9 instances with the same combination of $\{|V|, |K|\}$. For each method, the columns have the following meaning: $gap\%$ is the percentage gap between the best integer solution and the best lower bound found during the search in the branch-and-bound tree; $t$ is the CPU time in seconds; $ttb$ is the CPU time in seconds needed to find the best feasible (integer) solution; $root\%$ is the percentage gap left at the root node of the branch-and-bound tree; $B\&Bn$ is the number of nodes visited in the branch-and-bound tree (value 1 represents the root node). Moreover, $\#opt$ is the number of instances (out of 9) optimally solved by both the methods whereas column $\Delta\%$ shows the percentage deviation of the branch-and-price solutions with respect to the branch-and-cut ones (a negative value means that the branch-and-price has found a better solution).

Table 1: `B&P` vs `MM-B&C`

| | | | `B&P` | | | | | | | `MM-B&C` | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\|V\|$ | $\|K\|$ | $\|F\|$ | $\Delta\%$ | $\#opt$ | $gap\%$ | $t$ | $ttb$ | $root\%$ | $B\&Bn$ | $\#opt$ | $gap\%$ | $t$ | $ttb$ | $root\%$ | $B\&Bn$ |
| 10 | 10 | 5 | 0.00 | 9 | 0.00 | 2 | 1 | 100.00 | 1 | 9 | 0.00 | 4 | 1 | 88.33 | 5379 |
| 10 | 20 | 9 | 0.00 | 9 | 0.00 | 24 | 22 | 99.51 | 6 | 9 | 0.00 | 455 | 15 | 85.83 | 220431 |
| 10 | 30 | 14 | 0.00 | 9 | 0.00 | 44 | 39 | 99.84 | 4 | 3 | 16.78 | 4831 | 198 | 75.26 | 1530617 |
| 25 | 10 | 5 | 0.00 | 9 | 0.00 | 5 | 3 | 100.00 | 1 | 9 | 0.00 | 5 | 3 | 94.56 | 393 |
| 25 | 20 | 9 | 0.00 | 9 | 0.00 | 68 | 55 | 99.81 | 6 | 3 | 8.43 | 4808 | 319 | 85.58 | 296341 |
| 25 | 30 | 14 | -1.48 | 9 | 0.00 | 176 | 152 | 99.90 | 4 | 3 | 22.71 | 4879 | 3647 | 72.44 | 140179 |
| 40 | 10 | 5 | 0.00 | 9 | 0.00 | 62 | 47 | 99.84 | 2 | 9 | 0.00 | 730 | 215 | 90.57 | 28166 |
| 40 | 20 | 9 | -0.28 | 9 | 0.00 | 436 | 346 | 99.99 | 1 | 2 | 12.38 | 5641 | 4066 | 84.60 | 101011 |
| 40 | 30 | 14 | -2.53 | 9 | 0.00 | 1166 | 1052 | 99.97 | 2 | 0 | 26.92 | 7200 | 5076 | 71.93 | 139385 |
| | | | -0.48 | | 0.00 | 220 | 191 | 99.87 | 3 | | 9.69 | 3172 | 1505 | 83.23 | 273545 |

Looking at Table 1 results, we can notice that the branch-and-cut method is able to prove

optimality only for 47 instances out of 81. Moreover, it shows very high percentage gaps especially for instances with a higher number of products, and in general, as expected, for all instances with a larger fleet size. On the contrary, in all the instances, the branch-and-price always finds the optimal solution within the threshold time of two hours. Apart from the optimality gaps, the branch-and-price strictly improves the branch-and-cut solution in 8 instances, yielding an average percentage deviation ($\Delta\%$) of -0.48. Concerning computational times, the branch-and-cut results quite efficient for instances with a small fleet size, reaching the time limit of 2 hours in the remaining cases. B&P almost reaches the time limit only for one instance, but never exceeds 900 seconds in the remaining ones. Finally, the average CPU time ($t$) and the average time-to-best ($ttb$) for the branch-and-price are an order of magnitude lower than the competitor's ones.

Finally, to deeply understand the behavior of our B&P approach, in Tables 2–4 we provide additional statistics showing the detailed results obtained by the method for the instances with $|V| = 10$, 25 and 40, respectively. In each table, instances are uniquely identified by a combination of the tuple $\{|V|, |K|, f\%, Q, |F|\}$. Apart from the column headers already explained, $t_{UB}$ is the CPU time of the initial heuristic in seconds, $\#iter$ is the number of iterations performed by the column generation, $t_{sub}$ is the average CPU time in seconds needed for the solution of a subproblem, and $t_{root}$ is the CPU time needed to calculate the linear relaxation at the root node of the branch-and-bound tree.

Table 2: B&P details for $|V| = 10$ instances

| | | | | | B&P | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\|V\|$ | $\|K\|$ | $f\%$ | $Q$ | $\|F\|$ | $gap\%$ | $t_{UB}$ | $t$ | $\#iter$ | $t_{sub}$ | $\#col$ | $root\%$ | $t_{root}$ | $B\&Bn$ |
| 10 | 10 | 30 | 6 | 7 | 0.00 | 0.02 | 0.76 | 4 | 0.15 | 65 | 100.00 | 0.76 | 1 |
| 10 | 10 | 30 | 10 | 7 | 0.00 | 0.00 | 0.78 | 4 | 0.16 | 70 | 100.00 | 0.78 | 1 |
| 10 | 10 | 30 | 15 | 7 | 0.00 | 0.00 | 1.33 | 6 | 0.18 | 65 | 100.00 | 1.33 | 1 |
| 10 | 10 | 50 | 6 | 5 | 0.00 | 0.02 | 1.23 | 6 | 0.17 | 63 | 100.00 | 1.23 | 1 |
| 10 | 10 | 50 | 10 | 5 | 0.00 | 0.00 | 1.14 | 6 | 0.15 | 65 | 100.00 | 1.14 | 1 |
| 10 | 10 | 50 | 15 | 5 | 0.00 | 0.00 | 0.97 | 5 | 0.16 | 60 | 100.00 | 0.97 | 1 |
| 10 | 10 | 80 | 6 | 2 | 0.00 | 0.02 | 3.31 | 20 | 0.12 | 493 | 100.00 | 3.31 | 1 |
| 10 | 10 | 80 | 10 | 2 | 0.00 | 0.02 | 3.85 | 19 | 0.15 | 438 | 100.00 | 3.85 | 1 |
| 10 | 10 | 80 | 15 | 2 | 0.00 | 0.00 | 3.34 | 18 | 0.14 | 463 | 100.00 | 3.34 | 1 |
| 10 | 20 | 30 | 6 | 14 | 0.00 | 0.02 | 9.38 | 21 | 0.41 | 153 | 100.00 | 9.38 | 1 |
| 10 | 20 | 30 | 10 | 14 | 0.00 | 0.00 | 9.22 | 20 | 0.43 | 164 | 100.00 | 9.22 | 1 |
| 10 | 20 | 30 | 15 | 14 | 0.00 | 0.00 | 9.30 | 21 | 0.41 | 151 | 100.00 | 9.30 | 1 |
| 10 | 20 | 50 | 6 | 10 | 0.00 | 0.00 | 66.48 | 119 | 0.51 | 312 | 97.26 | 15.56 | 31 |
| 10 | 20 | 50 | 10 | 10 | 0.00 | 0.00 | 11.89 | 27 | 0.41 | 110 | 100.00 | 11.33 | 3 |
| 10 | 20 | 50 | 15 | 10 | 0.00 | 0.02 | 17.46 | 36 | 0.45 | 137 | 100.00 | 10.51 | 5 |
| 10 | 20 | 80 | 6 | 4 | 0.00 | 0.02 | 28.08 | 53 | 0.49 | 228 | 98.70 | 16.60 | 3 |
| 10 | 20 | 80 | 10 | 4 | 0.00 | 0.02 | 41.43 | 77 | 0.49 | 270 | 99.63 | 16.71 | 9 |
| 10 | 20 | 80 | 15 | 4 | 0.00 | 0.02 | 23.10 | 48 | 0.44 | 210 | 100.00 | 20.86 | 3 |
| 10 | 30 | 30 | 6 | 21 | 0.00 | 0.00 | 18.52 | 29 | 0.60 | 155 | 100.00 | 18.52 | 1 |
| 10 | 30 | 30 | 10 | 21 | 0.00 | 0.00 | 20.17 | 31 | 0.61 | 209 | 100.00 | 18.10 | 3 |
| 10 | 30 | 30 | 15 | 21 | 0.00 | 0.02 | 22.48 | 34 | 0.62 | 160 | 100.00 | 16.38 | 5 |
| 10 | 30 | 50 | 6 | 15 | 0.00 | 0.02 | 24.34 | 37 | 0.62 | 181 | 100.00 | 24.34 | 1 |
| 10 | 30 | 50 | 10 | 15 | 0.00 | 0.00 | 24.91 | 38 | 0.62 | 188 | 100.00 | 24.91 | 1 |
| 10 | 30 | 50 | 15 | 15 | 0.00 | 0.00 | 23.24 | 37 | 0.59 | 155 | 100.00 | 23.24 | 1 |
| 10 | 30 | 80 | 6 | 6 | 0.00 | 0.02 | 145.66 | 207 | 0.64 | 516 | 98.58 | 27.39 | 25 |
| 10 | 30 | 80 | 10 | 6 | 0.00 | 0.02 | 36.05 | 52 | 0.65 | 195 | 100.00 | 36.05 | 1 |
| 10 | 30 | 80 | 15 | 6 | 0.00 | 0.00 | 76.47 | 103 | 0.69 | 271 | 100.00 | 76.47 | 1 |
| | | | | | 0.00 | 0.01 | 23.14 | 39.93 | 0.41 | 205.44 | 99.78 | 14.87 | 3.89 |

Table 3: B&P details for $|V| = 25$ instances

| | | | | | B&P | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\|V\|$ | $\|K\|$ | $f\%$ | $Q$ | $\|F\|$ | $gap\%$ | $t_{UB}$ | $t$ | $\#iter$ | $t_{sub}$ | $\#col$ | $root\%$ | $t_{root}$ | $B\&Bn$ |
| 25 | 10 | 30 | 6 | 7 | 0.00 | 0.00 | 1.64 | 8 | 0.17 | 126 | 100.00 | 1.64 | 1 |
| 25 | 10 | 30 | 10 | 7 | 0.00 | 0.00 | 1.47 | 7 | 0.17 | 106 | 100.00 | 1.47 | 1 |
| 25 | 10 | 30 | 15 | 7 | 0.00 | 0.00 | 2.40 | 9 | 0.23 | 128 | 100.00 | 2.40 | 1 |
| 25 | 10 | 50 | 6 | 5 | 0.00 | 0.00 | 3.92 | 12 | 0.29 | 224 | 100.00 | 3.92 | 1 |
| 25 | 10 | 50 | 10 | 5 | 0.00 | 0.00 | 4.52 | 12 | 0.34 | 208 | 100.00 | 4.52 | 1 |
| 25 | 10 | 50 | 15 | 5 | 0.00 | 0.02 | 3.67 | 11 | 0.29 | 226 | 100.00 | 3.67 | 1 |
| 25 | 10 | 80 | 6 | 2 | 0.00 | 0.00 | 8.69 | 13 | 0.64 | 152 | 100.00 | 8.69 | 1 |
| 25 | 10 | 80 | 10 | 2 | 0.00 | 0.00 | 6.43 | 11 | 0.55 | 119 | 100.00 | 6.43 | 1 |
| 25 | 10 | 80 | 15 | 2 | 0.00 | 0.02 | 11.73 | 14 | 0.80 | 90 | 100.00 | 11.73 | 1 |
| 25 | 20 | 30 | 6 | 14 | 0.00 | 0.02 | 21.73 | 18 | 1.17 | 144 | 100.00 | 21.73 | 1 |
| 25 | 20 | 30 | 10 | 14 | 0.00 | 0.02 | 27.24 | 24 | 1.10 | 129 | 100.00 | 27.24 | 1 |
| 25 | 20 | 30 | 15 | 14 | 0.00 | 0.00 | 27.21 | 21 | 1.26 | 132 | 100.00 | 27.21 | 1 |
| 25 | 20 | 50 | 6 | 10 | 0.00 | 0.02 | 35.62 | 25 | 1.38 | 265 | 100.00 | 31.26 | 3 |
| 25 | 20 | 50 | 10 | 10 | 0.00 | 0.02 | 31.36 | 22 | 1.38 | 201 | 100.00 | 24.01 | 3 |
| 25 | 20 | 50 | 15 | 10 | 0.00 | 0.00 | 31.25 | 23 | 1.32 | 179 | 100.00 | 28.47 | 3 |
| 25 | 20 | 80 | 6 | 4 | 0.00 | 0.02 | 255.67 | 169 | 1.46 | 445 | 98.29 | 54.85 | 39 |
| 25 | 20 | 80 | 10 | 4 | 0.00 | 0.02 | 100.01 | 41 | 2.39 | 212 | 100.00 | 100.01 | 1 |
| 25 | 20 | 80 | 15 | 4 | 0.00 | 0.02 | 81.11 | 31 | 2.57 | 275 | 100.00 | 74.66 | 3 |
| 25 | 30 | 30 | 6 | 21 | 0.00 | 0.02 | 70.56 | 26 | 2.67 | 154 | 100.00 | 70.56 | 1 |
| 25 | 30 | 30 | 10 | 21 | 0.00 | 0.02 | 64.74 | 23 | 2.77 | 180 | 100.00 | 58.28 | 3 |
| 25 | 30 | 30 | 15 | 21 | 0.00 | 0.02 | 93.72 | 31 | 2.98 | 167 | 100.00 | 80.42 | 3 |
| 25 | 30 | 50 | 6 | 15 | 0.00 | 0.00 | 122.02 | 49 | 2.45 | 193 | 100.00 | 92.60 | 7 |
| 25 | 30 | 50 | 10 | 15 | 0.00 | 0.02 | 133.91 | 48 | 2.74 | 230 | 100.00 | 97.25 | 5 |
| 25 | 30 | 50 | 15 | 15 | 0.00 | 0.02 | 91.37 | 32 | 2.81 | 193 | 100.00 | 91.37 | 1 |
| 25 | 30 | 80 | 6 | 6 | 0.00 | 0.02 | 174.72 | 70 | 2.44 | 298 | 99.53 | 116.03 | 7 |
| 25 | 30 | 80 | 10 | 6 | 0.00 | 0.02 | 209.20 | 69 | 2.97 | 317 | 100.00 | 166.92 | 3 |
| 25 | 30 | 80 | 15 | 6 | 0.00 | 0.02 | 626.11 | 159 | 3.85 | 677 | 99.54 | 247.45 | 9 |
| | | | | | 0.00 | 0.01 | 83.04 | 36.22 | 1.60 | 213.70 | 99.90 | 53.88 | 3.81 |

Table 4: B&P details for $|V| = 40$ instances

| | | | | | B&P | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\|V\|$ | $\|K\|$ | $f\%$ | $Q$ | $\|F\|$ | $gap\%$ | $t_{UB}$ | $t$ | $\#iter$ | $t_{sub}$ | $\#col$ | $root\%$ | $t_{root}$ | $B\&Bn$ |
| 40 | 10 | 30 | 6 | 7 | 0.00 | 0.00 | 23.01 | 12 | 1.88 | 117 | 100.00 | 23.01 | 1 |
| 40 | 10 | 30 | 10 | 7 | 0.00 | 0.02 | 21.90 | 11 | 1.95 | 114 | 100.00 | 21.90 | 1 |
| 40 | 10 | 30 | 15 | 7 | 0.00 | 0.00 | 25.41 | 13 | 1.92 | 101 | 100.00 | 25.41 | 1 |
| 40 | 10 | 50 | 6 | 5 | 0.00 | 0.02 | 21.17 | 12 | 1.73 | 105 | 100.00 | 21.17 | 1 |
| 40 | 10 | 50 | 10 | 5 | 0.00 | 0.02 | 21.20 | 13 | 1.59 | 114 | 100.00 | 21.20 | 1 |
| 40 | 10 | 50 | 15 | 5 | 0.00 | 0.02 | 28.94 | 15 | 1.89 | 119 | 100.00 | 28.94 | 1 |
| 40 | 10 | 80 | 6 | 2 | 0.00 | 0.02 | 246.39 | 40 | 6.12 | 159 | 98.56 | 64.68 | 7 |
| 40 | 10 | 80 | 10 | 2 | 0.00 | 0.00 | 77.08 | 13 | 5.89 | 74 | 100.00 | 77.08 | 1 |
| 40 | 10 | 80 | 15 | 2 | 0.00 | 0.00 | 96.35 | 14 | 6.85 | 55 | 100.00 | 96.35 | 1 |
| 40 | 20 | 30 | 6 | 14 | 0.00 | 0.02 | 177.97 | 20 | 8.85 | 188 | 100.00 | 177.97 | 1 |
| 40 | 20 | 30 | 10 | 14 | 0.00 | 0.02 | 191.23 | 18 | 10.58 | 167 | 100.00 | 191.23 | 1 |
| 40 | 20 | 30 | 15 | 14 | 0.00 | 0.00 | 151.23 | 17 | 8.85 | 186 | 100.00 | 151.23 | 1 |
| 40 | 20 | 50 | 6 | 10 | 0.00 | 0.00 | 267.34 | 24 | 11.09 | 232 | 100.00 | 267.34 | 1 |
| 40 | 20 | 50 | 10 | 10 | 0.00 | 0.02 | 654.30 | 25 | 26.12 | 276 | 100.00 | 654.30 | 1 |
| 40 | 20 | 50 | 15 | 10 | 0.00 | 0.00 | 529.73 | 23 | 22.99 | 219 | 100.00 | 529.73 | 1 |
| 40 | 20 | 80 | 6 | 4 | 0.00 | 0.02 | 355.34 | 40 | 8.82 | 664 | 100.00 | 355.34 | 1 |
| 40 | 20 | 80 | 10 | 4 | 0.00 | 0.00 | 789.64 | 49 | 16.06 | 377 | 99.95 | 535.74 | 3 |
| 40 | 20 | 80 | 15 | 4 | 0.00 | 0.00 | 808.02 | 32 | 25.20 | 372 | 100.00 | 808.00 | 1 |
| 40 | 30 | 30 | 6 | 21 | 0.00 | 0.02 | 271.35 | 38 | 7.09 | 221 | 100.00 | 271.35 | 1 |
| 40 | 30 | 30 | 10 | 21 | 0.00 | 0.00 | 292.94 | 40 | 7.27 | 260 | 100.00 | 292.94 | 1 |
| 40 | 30 | 30 | 15 | 21 | 0.00 | 0.02 | 285.47 | 36 | 7.87 | 285 | 100.00 | 285.47 | 1 |
| 40 | 30 | 50 | 6 | 15 | 0.00 | 0.02 | 267.65 | 39 | 6.81 | 231 | 100.00 | 267.63 | 1 |
| 40 | 30 | 50 | 10 | 15 | 0.00 | 0.00 | 534.40 | 54 | 9.84 | 261 | 100.00 | 534.40 | 1 |
| 40 | 30 | 50 | 15 | 15 | 0.00 | 0.02 | 432.51 | 46 | 9.35 | 281 | 100.00 | 380.53 | 3 |
| 40 | 30 | 80 | 6 | 6 | 0.00 | 0.02 | 402.81 | 55 | 7.26 | 417 | 100.00 | 402.81 | 1 |
| 40 | 30 | 80 | 10 | 6 | 0.00 | 0.02 | 869.55 | 72 | 12.01 | 432 | 100.00 | 869.55 | 1 |
| 40 | 30 | 80 | 15 | 6 | 0.00 | 0.02 | 7141.03 | 181 | 39.30 | 977 | 99.72 | 1634.46 | 7 |
| | | | | | 0.00 | 0.01 | 554.96 | 35.26 | 10.19 | 259.41 | 99.93 | 332.95 | 1.59 |

It clearly seems that the superiority of the branch-and-price approach (in terms of solution quality and computational times) relies on the strength of the set-covering formulation used. In fact, in the most part of the instances, its linear relaxation already coincides with the optimal integer solution (see all the values equal to 100.00 in the $root\%$ column). In the remaining cases, when branching is needed, the branch-and-price tree never exceeds 40 nodes. On average, branch-and-price trees have 3 nodes, whereas the branch-and-cut trees have about 270000 nodes.

## 4.3 Extensive experiments

Being established that our new branch-and-price is the preferable solving method for the MVTPP-PIC with unitary demands (especially when the fleet size is quite large, as in practical applications), we define 72 new instances to better evaluate its performance. We consider a number of nodes (suppliers plus the depot) $|V| = \{20, 35, 50\}$, a number of products $|K| = \{10, 30, 50, 70\}$, and a percentage value of free product $f\% := \{30, 55, 80\}$. For each combination of $\{|V|, |K|, f\%\}$, two particular values of the capacity $Q := \{Q_1, Q_2\}$ are considered. In particular, $Q_1 := \lceil |K|/max\{|K_{tot}|, 2\} \rceil$ where $K_{tot}$ represents the set of totally incompatible products, and $Q_2 := \lceil 10Q_1/f\% \rceil$. The rationale of this choice is as follows. A larger vehicle capacity as $Q_1$ tends to generate instances where the number of vehicles needed to satisfy the demand is strongly (or solely) influenced by incompatibilities among products. The value $Q_2$ leads instead to more strict capacity constraints, and the final number of vehicle used will depend from the trade-off between vehicle capacity and incompatibilities. This new set of hard-to-solve problems also provides a test bed for future contributions and can be downloaded from the web page *http://www.ing.unibs.it/∼orgroup/instances.html*.

Tables 5–7 present the results for $|V| = 20$, 35 and 50 instances, respectively. Columns meaning has been already explained, apart from $\#v$ and $obj$ that represent the number of vehicles used and the objective function value of the best solution found. In column $gap\%$, values are highlighted in bold font when they correspond to a solution proved to be optimal. In this new experiments, the value of the parameters $negLabel_{LIM}$ and $allLabel_{LIM}$ have been doubled, being set to 100 and 1000, respectively. Finally, in order to avoid a useless explosion of the branch-and-bound tree, we also stop the branch-and-price when the percentage gap is less than 0.2%.

The new results confirm the overall goodness of our approach, pointing out also some of its limitations. The method works very well in particular for instances with $|V| = 20$ and $|V| = 35$ (see Tables 5 and 6). For 11 out of 72 instances the method is not able to terminate the column generation procedure at the root node within the threshold time (see entries with value 1 in column $B\&Bn$ and $gap\%$ greater than zero). In these cases the integrality gaps are quite high, because the restricted master heuristic seems quite ineffective and the best

Table 5: B&P results for $|V| = 20$ instances

| | | | | B&P | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\|K\|$ | $f\%$ | $Q$ | $\|F\|$ | $obj$ | $\#v$ | $gap\%$ | $t$ | $ttb$ | $\#iter$ | $t_{sub}$ | $\#col$ | $root\%$ | $t_{root}$ | $B\&Bn$ |
| 10 | 30 | 2 | 7 | 4267 | 6 | **0.00** | 14 | 0 | 36 | 0.28 | 82 | 98.43 | 1.14 | 17 |
| 10 | 30 | 5 | 7 | 3286 | 4 | **0.00** | 9 | 7 | 12 | 0.66 | 143 | 100.00 | 8.73 | 1 |
| 10 | 55 | 2 | 5 | 4036 | 5 | **0.00** | 1 | 1 | 5 | 0.19 | 57 | 100.00 | 1.24 | 1 |
| 10 | 55 | 5 | 5 | 2784 | 3 | **0.00** | 7 | 6 | 10 | 0.67 | 104 | 100.00 | 7.28 | 1 |
| 10 | 80 | 2 | 5 | 3928 | 5 | **0.00** | 6 | 0 | 15 | 0.31 | 77 | 97.58 | 0.81 | 9 |
| 10 | 80 | 5 | 2 | 2422 | 2 | **0.00** | 13 | 11 | 19 | 0.62 | 128 | 100.00 | 12.95 | 1 |
| 30 | 30 | 5 | 21 | 8419 | 10 | **0.00** | 55 | 52 | 41 | 1.26 | 235 | 100.00 | 40.30 | 5 |
| 30 | 30 | 15 | 21 | 8419 | 10 | **0.00** | 57 | 54 | 43 | 1.24 | 238 | 100.00 | 51.48 | 3 |
| 30 | 55 | 3 | 14 | 7899 | 10 | **0.00** | 62 | 48 | 65 | 0.86 | 222 | 99.68 | 36.42 | 7 |
| 30 | 55 | 15 | 14 | 6359 | 7 | **0.00** | 70 | 66 | 45 | 1.46 | 300 | 100.00 | 69.80 | 1 |
| 30 | 80 | 2 | 15 | 10521 | 15 | **0.00** | 77 | 29 | 104 | 0.65 | 197 | 99.33 | 20.83 | 35 |
| 30 | 80 | 15 | 6 | 4655 | 4 | **0.00** | 201 | 172 | 103 | 1.82 | 568 | 99.88 | 146.77 | 5 |
| 50 | 30 | 9 | 35 | 11266 | 15 | 0.19 | 3553 | 337 | 878 | 3.64 | 978 | 99.30 | 245.46 | 687 |
| 50 | 30 | 25 | 35 | 11266 | 15 | 0.07 | 4418 | 449 | 1056 | 3.80 | 1050 | 99.30 | 236.43 | 879 |
| 50 | 55 | 4 | 23 | 10526 | 13 | **0.00** | 1605 | 629 | 538 | 2.66 | 814 | 99.54 | 136.35 | 287 |
| 50 | 55 | 17 | 23 | 9393 | 12 | **0.00** | 341 | 309 | 90 | 3.65 | 521 | 99.12 | 305.21 | 3 |
| 50 | 80 | 4 | 13 | 9615 | 13 | **0.00** | 365 | 360 | 160 | 1.94 | 576 | 98.16 | 151.15 | 33 |
| 50 | 80 | 25 | 10 | 6181 | 5 | **0.00** | 618 | 613 | 155 | 3.67 | 1014 | 100.00 | 617.78 | 1 |
| 70 | 30 | 6 | 49 | 19536 | 21 | **0.00** | 589 | 561 | 98 | 5.77 | 595 | 100.00 | 441.22 | 9 |
| 70 | 30 | 18 | 49 | 19460 | 21 | **0.00** | 658 | 651 | 99 | 6.43 | 529 | 100.00 | 470.68 | 13 |
| 70 | 55 | 7 | 32 | 14209 | 14 | **0.00** | 1574 | 1534 | 243 | 5.84 | 1119 | 99.75 | 447.00 | 33 |
| 70 | 55 | 35 | 32 | 14133 | 14 | **0.00** | 1037 | 1021 | 182 | 5.16 | 1043 | 100.00 | 624.95 | 21 |
| 70 | 80 | 5 | 14 | 13101 | 14 | **0.00** | 3202 | 3048 | 279 | 3.13 | 1170 | 99.45 | 1046.77 | 33 |
| 70 | 80 | 35 | 14 | 9480 | 8 | **0.00** | 2331 | 2305 | 324 | 5.90 | 2426 | 100.00 | 2143.90 | 9 |
| | | | | | | 0.01 | 869 | 511 | 192 | 2.57 | 591 | 99.56 | 302.69 | 87 |

Table 6: B&P results for $|V| = 35$ instances

| | | | | B&P | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\|K\|$ | $f\%$ | $Q$ | $\|F\|$ | $obj$ | $\#v$ | $gap\%$ | $t$ | $ttb$ | $\#iter$ | $t_{sub}$ | $\#col$ | $root\%$ | $t_{root}$ | $B\&Bn$ |
| 10 | 30 | 2 | 7 | 1596 | 6 | **0.00** | 6 | 1 | 11 | 0.49 | 75 | 98.31 | 2.65 | 3 |
| 10 | 30 | 5 | 7 | 1351 | 3 | **0.00** | 12 | 10 | 14 | 0.76 | 71 | 100.00 | 11.51 | 1 |
| 10 | 55 | 2 | 5 | 1538 | 5 | **0.00** | 7 | 0 | 12 | 0.49 | 78 | 98.24 | 2.42 | 3 |
| 10 | 55 | 4 | 5 | 1342 | 4 | **0.00** | 9 | 2 | 11 | 0.75 | 64 | 100.00 | 8.89 | 1 |
| 10 | 80 | 2 | 5 | 1538 | 5 | **0.00** | 5 | 0 | 10 | 0.44 | 80 | 94.86 | 2.40 | 3 |
| 10 | 80 | 5 | 2 | 1187 | 2 | **0.00** | 14 | 9 | 14 | 0.91 | 104 | 100.00 | 13.63 | 1 |
| 30 | 30 | 4 | 21 | 5245 | 12 | **0.00** | 111 | 68 | 26 | 4.16 | 263 | 99.96 | 99.76 | 3 |
| 30 | 30 | 10 | 21 | 5236 | 12 | **0.00** | 137 | 113 | 28 | 4.79 | 256 | 100.00 | 107.94 | 3 |
| 30 | 55 | 3 | 14 | 4703 | 11 | **0.00** | 155 | 59 | 50 | 2.98 | 395 | 99.46 | 80.92 | 7 |
| 30 | 55 | 15 | 14 | 4097 | 7 | **0.00** | 500 | 452 | 45 | 11.00 | 368 | 100.00 | 404.31 | 3 |
| 30 | 80 | 2 | 15 | 5713 | 15 | **0.00** | 73 | 25 | 47 | 1.46 | 242 | 98.79 | 28.13 | 13 |
| 30 | 80 | 8 | 6 | 3501 | 5 | **0.00** | 1606 | 1109 | 152 | 10.38 | 562 | 99.64 | 558.62 | 19 |
| 50 | 30 | 6 | 35 | 12907 | 20 | **0.00** | 941 | 905 | 47 | 19.90 | 368 | 100.00 | 941.15 | 1 |
| 50 | 30 | 17 | 35 | 12907 | 20 | **0.00** | 2907 | 2421 | 57 | 50.83 | 475 | 100.00 | 2906.64 | 1 |
| 50 | 55 | 3 | 23 | 11160 | 17 | **0.00** | 925 | 788 | 88 | 10.31 | 494 | 99.24 | 246.75 | 25 |
| 50 | 55 | 13 | 23 | 10988 | 13 | 21.81 | 7200 | 7185 | 72 | 99.69 | 863 | - | - | 1 |
| 50 | 80 | 4 | 13 | 8723 | 13 | **0.00** | 827 | 813 | 149 | 5.24 | 578 | 97.79 | 430.89 | 19 |
| 50 | 80 | 25 | 10 | 11182 | 6 | 54.34 | 7200 | 0 | 111 | 64.45 | 1135 | - | - | 1 |
| 70 | 30 | 6 | 49 | 12308 | 25 | **0.00** | 2041 | 2002 | 120 | 16.68 | 812 | 99.89 | 1092.00 | 27 |
| 70 | 30 | 18 | 49 | 12133 | 25 | **0.00** | 1783 | 1732 | 89 | 19.76 | 744 | 100.00 | 1679.37 | 5 |
| 70 | 55 | 5 | 32 | 10195 | 16 | **0.00** | 2743 | 2200 | 175 | 15.02 | 763 | 99.83 | 1134.65 | 27 |
| 70 | 55 | 24 | 32 | 9521 | 16 | **0.00** | 4463 | 4399 | 163 | 26.78 | 1215 | 100.00 | 4463.18 | 1 |
| 70 | 80 | 5 | 14 | 8944 | 14 | **0.00** | 4901 | 4866 | 275 | 10.05 | 1167 | 99.42 | 3583.14 | 35 |
| 70 | 80 | 35 | 14 | 15529 | 9 | 74.08 | 7200 | 0 | 110 | 65.04 | 1203 | - | - | 1 |
| | | | | | | 6.26 | 1907 | 1215 | 78 | 18.43 | 516 | 99.31 | 847.57 | 9 |

solutions are compared to very poor lower bounds (computed as the maximum between the value obtained by the formula (35) and the optimal solution value of the LP relaxation of (1)–(12) without constraints (4)). However, the B&P has been able to solve optimally about 78% of the instances (in particular, it has found 22, 21 and 13 optimal solutions out of

Table 7: B&P results for $|V| = 50$ instances

| | | | | B&P | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\|K\|$ | $f\%$ | $Q$ | $\|F\|$ | $obj$ | $\#v$ | $gap\%$ | $t$ | $ttb$ | $\#iter$ | $t_{sub}$ | $\#col$ | $root\%$ | $t_{root}$ | $B\&Bn$ |
| 10 | 30 | 2 | 7 | 3045 | 6 | **0.00** | 9 | 3 | 8 | 1.03 | 63 | 100.00 | 8.66 | 1 |
| 10 | 30 | 4 | 7 | 2869 | 6 | **0.00** | 36 | 20 | 11 | 3.23 | 104 | 100.00 | 36.21 | 1 |
| 10 | 55 | 2 | 5 | 2803 | 5 | **0.00** | 13 | 5 | 10 | 1.25 | 70 | 100.00 | 13.10 | 1 |
| 10 | 55 | 4 | 5 | 2371 | 4 | **0.00** | 70 | 35 | 12 | 5.76 | 168 | 100.00 | 69.94 | 1 |
| 10 | 80 | 2 | 5 | 2803 | 5 | **0.00** | 15 | 9 | 12 | 1.17 | 76 | 100.00 | 14.73 | 1 |
| 10 | 80 | 5 | 2 | 1779 | 2 | **0.00** | 147 | 111 | 17 | 8.56 | 179 | 100.00 | 146.69 | 1 |
| 30 | 30 | 3 | 21 | 5803 | 12 | **0.00** | 291 | 180 | 32 | 9.00 | 243 | 100.00 | 291.10 | 1 |
| 30 | 30 | 8 | 21 | 5708 | 11 | **0.00** | 4651 | 3829 | 38 | 122.29 | 257 | 100.00 | 3574.22 | 3 |
| 30 | 55 | 3 | 14 | 5487 | 10 | **0.00** | 490 | 437 | 48 | 10.09 | 297 | 99.48 | 235.78 | 11 |
| 30 | 55 | 15 | 14 | 13662 | 8 | 89.22 | 7200 | 2 | 22 | 327.19 | 280 | - | - | 1 |
| 30 | 80 | 2 | 15 | 6448 | 15 | **0.00** | 197 | 159 | 56 | 3.14 | 2036 | 99.58 | 132.74 | 5 |
| 30 | 80 | 15 | 6 | 9273 | 4 | 84.12 | 7200 | 0 | 12 | 599.95 | 207 | - | - | 1 |
| 50 | 30 | 9 | 35 | 7919 | 14 | **0.00** | 5117 | 4468 | 67 | 76.16 | 521 | 100.00 | 5116.93 | 1 |
| 50 | 30 | 25 | 35 | 7919 | 14 | **0.00** | 4144 | 3832 | 56 | 73.79 | 504 | 100.00 | 4143.51 | 1 |
| 50 | 55 | 5 | 23 | 7271 | 12 | 0.15 | 7200 | 3237 | 158 | 45.22 | 734 | 99.28 | 2587.21 | 30 |
| 50 | 55 | 25 | 23 | 6808 | 11 | 2.04 | 7200 | 7124 | 61 | 117.82 | 644 | - | - | 1 |
| 50 | 80 | 3 | 17 | 8425 | 17 | **0.00** | 2915 | 1817 | 186 | 14.96 | 719 | 98.42 | 673.80 | 61 |
| 50 | 80 | 17 | 10 | 13382 | 7 | 80.90 | 7200 | 0 | 45 | 159.82 | 603 | - | - | 1 |
| 70 | 30 | 12 | 49 | 29051 | 19 | 89.87 | 7200 | 2719 | 49 | 146.77 | 528 | - | - | 1 |
| 70 | 30 | 35 | 49 | 32186 | 19 | 90.86 | 7200 | 6159 | 41 | 175.46 | 480 | - | - | 1 |
| 70 | 55 | 5 | 32 | 9752 | 18 | 0.37 | 7200 | 6338 | 100 | 71.47 | 795 | 99.60 | 4845.81 | 5 |
| 70 | 55 | 24 | 32 | 25044 | 17 | 88.25 | 7200 | 0 | 33 | 218.00 | 410 | - | - | 1 |
| 70 | 80 | 5 | 14 | 9266 | 14 | 10.28 | 7200 | 5302 | 147 | 41.20 | 1033 | 89.72 | 6930.73 | 2 |
| 70 | 80 | 35 | 14 | 15780 | 8 | 81.35 | 7200 | 0 | 13 | 553.76 | 180 | - | - | 1 |
| | | | | | | 25.73 | 4054 | 1908 | 51 | 116.13 | 464 | 99.13 | 1801.32 | 6 |

24 instances with $|V| = 20$, 35, and 50, respectively). For the whole set of instances with $|V| = 20$, the B&P generates an average percentage gap equal to 0.01, and the average CPU time is about 870 seconds.

Some other trends can also be noticed. For example, the complexity of the instances grows not surprisingly with the number of products involved, but also with the percentage of free products. The reason might be that our subproblem solving procedures are much more efficient when products present a lot of incompatibilities. Moreover, instances with a vehicle capacity equal to $Q_1$ (even lines of each table) appear in general a bit harder to solve than the ones with capacity $Q_2$ (odd lines). It seems that a smaller value of $Q$, instead of enhancing the combined effect of the two types of restrictions (vehicle capacity and incompatibilities), mitigates the combinatorial hardness caused by the PIC.

In order to better understand the benefits of some components of our branch-and-price (B&P), we compare its performance to those obtained by the method excluding stabilization (B&P-NoStab) and the two-phase restricted master heuristic (B&P-NoRMH). Table 8 shows this comparison on a subset of 24 instances of the type $\{|V|, |K|, f\%, Q\}$, where $|V| = \{20, 35, 50\}$, $|K| = \{10, 70\}$, $f\% = \{30, 80\}$, and $Q = \{Q_1, Q_2\}$. Each table entry refers to the average result out of 4 instances with the same combination of $\{|V|, |K|\}$. We can see that B&P-NoRMH achieves 2 optimal solutions less and shows an optimality gap about the 5% worse than the standard version. This is due to a slower convergence of the column generation procedure. In fact, on average, the overall CPU time and that for solving the

linear relaxation at the root node are greater. This is not true for the easiest instances, where the stabilization seems unnecessary. On the contrary, B&P-NoRMH is slightly less time consuming with respect to the standard version (since it does not make use of the restricted master heuristic), but presents a worse optimality gap and a consistent average percentage deviation of the solutions with respect to the best found (around 10%). In conclusion, as expected, both the stabilization method and the restricted master heuristic implemented give efficiency and effectiveness to the solution procedure, in particular for the hardest instances.

Table 8: B&P vs B&P-NoStab and B&P-NoRMH

| | | B&P | | | | B&P-NoStab | | | | | B&P-NoRMH | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\|V\|$ | $\|K\|$ | $\#opt$ | $gap\%$ | $t$ | $t_{root}$ | $\#opt$ | $gap\%$ | $t$ | $t_{root}$ | $\Delta\%$ | $\#opt$ | $gap\%$ | $t$ | $t_{root}$ | $\Delta\%$ |
| 20 | 10 | 4 | 0.00 | 10 | 6 | 4 | 0.00 | 8 | 6 | 0.00 | 4 | 0.00 | 9 | 6 | 0.00 |
| 20 | 70 | 4 | 0.00 | 1695 | 1026 | 3 | 12.80 | 3708 | 2310 | 26.22 | 4 | 0.00 | 1103 | 770 | 0.00 |
| 35 | 10 | 4 | 0.00 | 9 | 8 | 4 | 0.00 | 11 | 10 | 0.00 | 4 | 0.00 | 9 | 7 | 0.00 |
| 35 | 70 | 3 | 18.52 | 3981 | 3389 | 3 | 17.95 | 4713 | 4128 | 0.00 | 3 | 17.37 | 4161 | 2922 | 0.00 |
| 50 | 10 | 4 | 0.00 | 52 | 52 | 4 | 0.00 | 42 | 42 | 0.00 | 4 | 0.00 | 51 | 51 | 0.00 |
| 50 | 70 | 0 | 68.09 | 7200 | 7133 | 0 | 83.65 | 7200 | 7200 | 56.60 | 0 | 83.64 | 7200 | 6732 | 56.60 |
| | | | 14.43 | 2158 | 1935 | | 19.07 | 2614 | 2283 | 13.80 | | 16.84 | 2089 | 1748 | 9.43 |

On the same subset of instances we now evaluate the contribution to the column generation procedure of the two solution methods proposed for the pricing problem, i.e., the labeling algorithm (Lab) and the tailored branch-and-cut (B&C). The average CPU time per iteration ($t_{SUB}$) of the two methods is presented in Figure 2, whereas Figure 3 shows the percentage number of iterations executed and the number of columns added by the two methods out of the total number. Results are grouped by value of $|V|$. In Figure 2 we see that $t_{SUB}$ for B&C

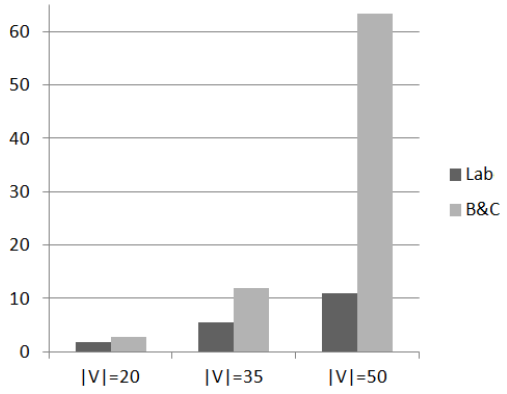



Figure 2: Average time per iteration for the two pricing problem solution methods.

grows exponentially with the number of nodes, whereas it grows quite linearly for Lab. This depends on the fact that the labeling algorithm, in our branch-and-price implementation, is stopped prematurely as soon as it results to be ineffective, whereas the branch-and-cut always needs to ensure optimality for the pricing problem. Figure 3 reveals instead how the two methods collaborate in the column generation procedure. Not surprisingly, the labeling

algorithm is always executed very few times with respect to the total number of iterations (this value never exceeds, on average, the 5%). However, in proportion, it generates a conspicuous part of the total number of columns (20% for $|V| = 35$ instances, and almost 35% for $|V| = 50$ instances). `B&C`, instead, returns the most part of the columns but using a number of iterations about 20 times larger. This means that the intrinsic drawbacks of the two pricing problem solution procedures are very well compensated by the use of the switching method exposed in Section 3.2.4.

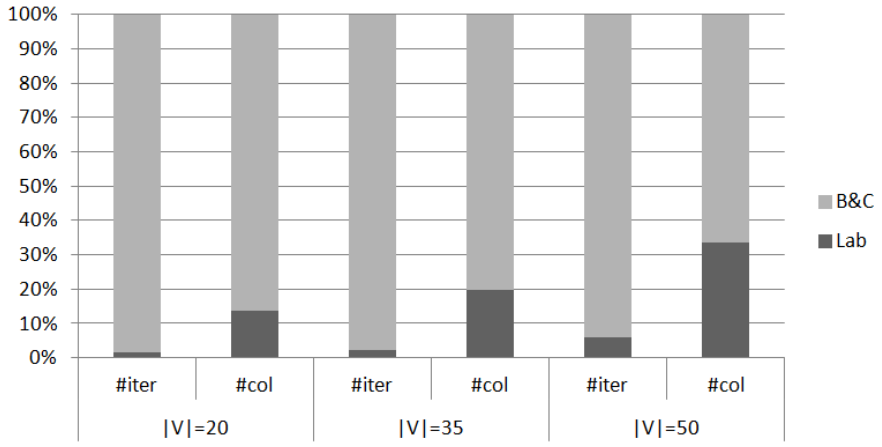



Figure 3: Percentage of iterations and columns for the two pricing problem solution methods.

# 5 Conclusions

In this paper we dealt with the Multi-Vehicle Travelling Purchaser Problem with Pairwise Incompatibility Constraints and unitary demands for the products. We developed a branch-and-price solution approach embedding a column generation procedure within a branch-and-bound framework. The column generation relies on a set-covering formulation of the problem where a column corresponds to a feasible route associated with a compatible purchasing plan. We proposed two exact methods to solve the pricing problem, i.e. a labeling algorithm solving a Resource Constrained Elementary Shortest Path Problem on an expanded graph and a tailored branch-and-cut approach. We also proposed an initial heuristic, three different branching rules, and several acceleration techniques to speed up the process. Experimental tests have shown that the resulting implementation outperforms the branch-and-cut method existing in the literature, both in solution quality and in computational time, especially when the size of the fleet involved is quite large as in real applications. Our branch-and-price has been able to optimally solve in a reasonable amount of time instances with up to 50 suppliers, 70 products, and 70% of cross-incompatibilities among products.

# Acknowledgements

This work was partially supported by the Canadian Natural Science and Engineering Research Council (NSERC) through its Discovery Grants Program.

# References

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., Vance, P. H., 1998. Branch-and-price: Column generation for solving huge integer programs. Operations Research 46 (3), 316–329.

Bianchessi, N., Mansini, R., Speranza, M., 2014. The distance constrained multiple vehicle traveling purchaser problem. European Journal of Operational Research 235 (1), 73–87.

Burstall, R., 1966. A heuristic method for a job-scheduling problem. Operational Research Quarterly 17 (3), 291–304.

Choi, M., Lee, S., 2011. The multiple traveling purchaser problem for maximizing system's reliability with budget constraints. Expert Systems with Applications 38, 9848–9853.

Christofides, N., Mingozzi, A., Toth, P., 1979. Loading problems. In: Christofides, N., Mingozzi, A., Sandi, C., Toth, P. (Eds.), Combinatorial Optimization. John Wiley, London, pp. 339–369.

Dantzig, G. B., Wolfe, P., 1960. Decomposition principle for linear programs. Operations Research 8 (1), 101–111.

Desaulniers, G., 2010. Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. Operations Research 58 (1), 179–192.

Desaulniers, G., Desrosiers, J., Solomon, M. M., 2002. Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. In: Essays and Surveys in Metaheuristics. Vol. 15 of Operations Research/Computer Science Interfaces Series. Springer US, pp. 309–324.

du Merle, O., Villeneuve, D., Desrosiers, J., Hansen, P., 1999. Stabilized column generation. Discrete Mathematics 194, 229–237.

Feillet, D., Dejax, P., Gendreau, M., 2005. The profitable arc tour problem: Solution with a branch-and-price algorithm. Transportation Science 39 (4), 539–552.

Fischetti, M., Salazar-González, J.-J., Toth, P., 1998. Solving the orienteering problem through branch-and-cut. INFORMS Journal on Computing 10, 133–148.

Gendreau, M., Dejax, P., Feillet, D., Gueguen, C., 2006. Vehicle routing with time windows and split deliveries. Technical report 2006-851, Laboratoire Informatique d'Avignon.

Gendreau, M., Laporte, G., Semet, F., 2004. Heuristics and lower bounds for the bin packing problem with conflicts. Computers & Operations Research 31 (3), 347–358.

Goossens, D., Spieksma, F. C., 2009. The transportation problem with exclusionary side constraints. 4OR 7 (1), 51–60.

Helsgaun, K., 2000. An effective implementation of the lin-kernighan traveling salesman heuristic. European Journal of Operational Research 126, 106–130.

Iori, M., Martello, S., 2010. Routing problems with loading constraints. TOP 18 (1), 4–27.

Irnich, S., Desaulniers, G., 2005. Shortest path problems with resource constraints. In: Desaulniers, G., Desrosiers, J., Solomon, M. M. (Eds.), Column Generation. Springer US, pp. 33–65.

Jin, M., Liu, K., Eksioglu, B., 2008. A column generation approach for the split delivery vehicle routing problem. Operations Research Letters 36 (2), 265–270.

Laporte, G., Riera-Ledesma, J., Salazar-González, J.-J., 2003. A branch-and-cut algorithm for the undirected traveling purchaser problem. Operations Research 51 (6), 940–951.

Manerba, D., Mansini, R., 2014. A branch-and-cut algorithm for the multi-vehicle traveling purchaser problem with pairwise incompatibility constraints. Networks (Accepted August 22, 2014. To appear).

Mansini, R., Tocchella, B., 2009. The traveling purchaser problem with budget constraint. Computers & Operations Research 36, 2263–2274.

Riera-Ledesma, J., Salazar-González, J.-J., 2012. Solving school bus routing using the multiple vehicle traveling purchaser problem: a branch-and-cut approach. Computers & Operations Research 39 (1), 391–404.

Riera-Ledesma, J., Salazar-González, J.-J., 2013. A column generation approach for a school bus routing problem with resource constraints. Computers & Operations Research 40 (1), 566–583.

Rousseau, L.-M., Gendreau, M., Feillet, D., 2007. Interior point stabilization for column generation. Operation Research Letters 35 (5), 660–668.

Sun, M., 2002. The transportation problem with exclusionary side constraints and two branch-and-bound algorithms. European Journal of Operational Research 140 (3), 629–647.

Voss, S., 1996. Dynamic tabu search strategies for the traveling purchaser problem. Annals of Operations Research 63 (2), 253–275.