A branch-and-cut algorithm for the Multi-Vehicle Traveling Purchaser Problem with Exclusionary Side Constraints

Daniele Manerba, Renata Mansini

Department of Information Engineering, University of Brescia, Italy

Operations Research Group

Technical Report No. 02, September 2013

Abstract

We introduce a problem where a fleet of vehicles is available to visit suppliers offering various products at different prices and quantities, with the aim to select a subset of suppliers so to satisfy products demand at the minimum traveling and purchasing costs. Vehicles have a predefined capacity and products can be incompatible so that their load on the same vehicle is forbidden. We call this problem the Multi-Vehicle Traveling Purchaser Problem with Exclusionary Side Constraints. We show how a three-index one-commodity flow formulation for the problem may be easy to implement with a common MILP solver, but highly non efficient when solving large size instances. We concentrate on a formulation using connectivity constraints to exclude subtours and introduce a branch-and-cut framework using a preprocessing routine and the separation of different valid inequalities. We also introduce a four-step heuristic based on the solution of different subproblems and used it to provide a initial feasible solution. A streamlined version of the proposed exact method tested on a large set of instances with up to 50 suppliers, 100 products and 20% of crossed incompatibility between products largely outperforms the plain resolution by the commercial solver Cplex 12.3. Also the heuristic approach comes out to be quite effective and extremely efficient.

Keywords: Traveling Purchaser Problem; Multi-vehicle; Exclusionary Side Constraints; branch-and-cut.

1 Introduction

Consider a depot 0, a set of suppliers (dispersed in a geographical area around the depot) $M := \{1, \ldots, m\}$ and a set of products $K := \{1, \ldots, n\}$. An integer positive demand d_k is associated to each product $k \in K$, that can be purchased 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 $G = (V, E)$ be a complete graph where $V := \{0\} \cup M$ is the vertex set and E is the edge set. For each edge $e \in E$, the traveling cost c_e is calculated as Euclidean distance. The well-known Traveling Purchaser Problem (TPP) looks for a cycle in G starting at and ending to the depot 0, and visiting a subset of suppliers so to exactly satisfy products demand, while minimizing both traveling and purchasing costs. Following the well-established literature we will refer to the special case of the problem with unlimited supplies, i.e. $q_{ik} \geq d_k$ for all $k \in K$ and $i \in M_k$, as the *unrestricted* TPP, and to the more general case as *restricted* (or *capacitated*) TPP. Notice that the unrestricted case is equivalent to assuming that $d_k = 1$ and $q_{ik} = 1$ for all $k \in K$ and $i \in M_k$.

The TPP was introduced in relation with job scheduling by Burstall [7], but since it has been presented in a routing context by Ramesh [22], the problem has gained a lot of attention in the specialized literature. There are several motivations behind the study of this problem. First of all, the TPP arises in many practical procurement contexts where the decision maker has to deal simultaneously with the suppliers selection, the purchasing decisions at the suppliers, and the routing optimization. Secondly, the problem is proved to be NP-hard and quite challenging to face in practice too, because of its complex combinatorial structure (see Laporte et al. [17]). Hence, especially in the last decade, both exact and heuristic methods have been proposed for the problem and its variants either in deterministic or in dynamic and stochastic contexts (see Mansini and Tocchella [19] for the TPP with budget constraint, Angelelli et al. [2] and Angelelli et al. [3] for a dynamic version with quantities decreasing over time, Angelelli et al. [4] for a time-dependent stochastic variant, Beraldi et al. [5] for the TPP under uncertainty, and Gouveia et al. [14] for the TPP with additional side-constraints).

Aside the basic version of the TPP, real and large-size procurement settings need to account for at least two more aspects to be correctly modeled. Both of them are related to the presence of real constraints. The first aspect concerns loading capacities or maximum service times. In practice, the problem optimization has to be done considering a fleet of vehicles and not just a single vehicle. Few contributions can be found in the literature about the multi-vehicle version of the TPP. In Choi and Lee [9] the authors introduce a reliability optimization problem as variant of the multi-vehicle TPP. In Riera-Ledesma and Salazar-Gonzàlez [23] the authors analyze the generalization of the asymmetric unrestricted TPP to

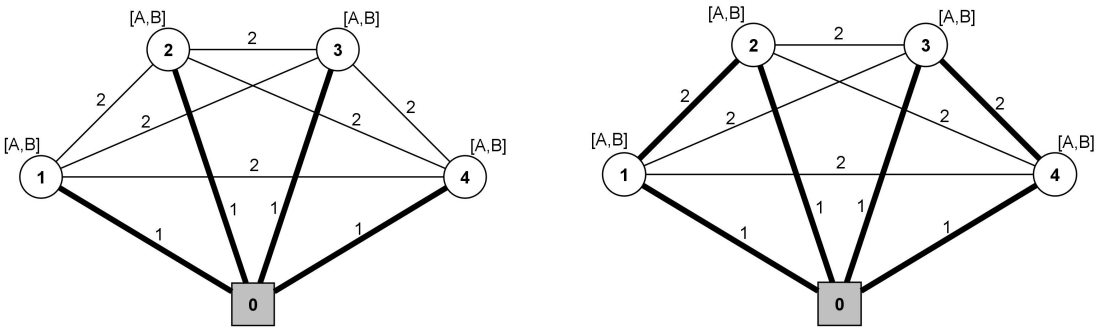
the multiple vehicle case with capacity constraint. The problem is described as a location routing problem in the context of school bus routing. The authors propose a model based on a single commodity flow formulation and provide valid inequalities to be used in a branch-and-cut algorithm tested on a large set of randomly generated instances. Very recently, the same authors propose a branch-and-cut-and-price approach to address a generalization of the problem taking into account bounds not only on the loading capacity but also on other resources (Riera-Ledesma and Salazar-Gonzàlez [24]). Finally, in Bianchessi et al. [6] the authors present different mathematical programming formulations for a distance constrained variant of the multi vehicle TPP and propose a branch-and-price algorithm to solve a set partitioning formulation where columns represent feasible routes for the vehicles.

The second aspect is related to the possibility that two or more products are incompatible, that is they could not be loaded on the same vehicle. Examples of incompatible products can be found in different real contexts: food goods cannot be mixed with chemicals, substances with hazardous properties may become unstable if mixed with other products and so on. Incompatibility constraints are usually treated in three different ways. The first solution implies the use of heterogeneous, so called *dedicated vehicles*. This trivially leads, if a lot of incompatibilities exist among products, to the need of an enormous fleet size and consequently to a great waste of resources and money. As an alternative the vehicles can make use of some separation devices. In literature the problem is known as multi-compartment vehicle routing problem: each customer may order several products, the vehicles are identical and have several compartments, and each compartment being dedicated to one product. The demand of each customer for a product must be entirely delivered by one single vehicle. Interest readers are referred to Fallahi et al. [11], Muyldermans and Pang [21] and to Mendoza et al. [20] for a stochastic variant. Vehicle routing problems with multi-compartment loading are also analyzed in the survey by Iori and Martello [16]. Finally, the last way to deal with products incompatibility is to optimize the vehicles routes avoiding to load incompatible products on the same vehicle. At this aim the so-called exclusionary side constraints are added to the problem formulation. In the literature different contributions can be found on the application of exclusionary side constraints to other problems (see for instance Goossens and Spieksma [13] and Cao and Uebe [8] for the Transportation Problem).

In this paper we generalize the restricted TPP by considering a fleet instead of a single vehicle (multi-vehicle variant) and by dealing with products that may be incompatibile (exclusionary side constraints). We call the problem the Multi Vehicle Traveling Purchaser Problem with Exclusionary Side Constraints (MVTPP-ESC). To define this generalization we need for additional notation with respect to the basic TPP. Let us consider a set $F := \{1, ..., u\}$ of homogeneous vehicles with equal capacity Q. Moreover let $B := \{(k, g) : k, g \in K\}$ be the set of pairs of products that cannot be shipped on the same vehicle simultaneously,

i.e. the set of exclusionary side constraints. The MVTPP-ESC aims at determining a simple cycle for each vehicle $v \in F$ passing through the depot and visiting a subset of suppliers so that the total demand is satisfied at minimum purchasing and traveling cost, while guaranteeing all exclusionary constraints and without exceeding vehicles capacity. It is easy to see that the MVTPP-ESC is an $\mathcal{NP}$-hard problem reducing to the TPP when $B = \emptyset$, $|F| = 1$ and $Q \geq \sum_{k \in K} d_k$. In order to satisfy incompatibilities the problem allows multiple visits to the same supplier. This aspect characterizes also the routing problems with split deliveries (see Dror and Trudeau [10]). However, while in the latter problem the split induces a traveling costs reduction in our case the split may cause a costs increase being forced by the exclusionary side constraints (see Example 1 where the introduction of products incompatibility doubles the traveling costs).

Example 1 *Let us consider 2 incompatible products, A and B, a depot* 0 *and* 4 *suppliers offering a single unit for each product. The demand for each product is* 4*, and the capacity of each vehicle is* 2*. The graph and the routing cost for each edge are shown in Figure 1. If we remove incompatibility between A and B the optimal solution is represented by four direct visits to the suppliers, as shown in Figure 1(a), where A and B are loaded on the same vehicle with a total traveling cost equal to* 8*. On the contrary, for the MVTPP-ESC, the optimal solution is represented by four tours (two for product A and two for product B) with a total traveling cost equal to* 16*. The two optimal tours for collecting A (and equivalently for collecting B) are highlighted in bold in Figure 1(b).*



(a) MVTPP solution

(b) MVTPP-ESC solution

Figure 1: Difference between MVTPP and MVTPP-ESC

The rest of the paper is organized as follows. In Section 2 we introduce two mathematical models for the problem, a three-index single-commodity flow formulation and a linear mixed integer formulation with connectivity constraints, and study different classes of valid inequalities. In Section 3 we introduce a branch-and-cut approach including a preprocessing routine and the separation of ad-hoc valid inequalities, while Section 4 describes a four-step

heuristic that is used to provide an initial feasible solution for the exact method. In Section 5 the results obtained on a large set of instances are analyzed. We test different variants of the proposed heuristic and compare the resulting streamlined version of our exact method to the performance of Cplex 12.3 when solving the three-index single-commodity flow formulation. Finally, in Section 6 some conclusions are drawn, and future developments are briefly sketched.

2 Mathematical formulations and valid inequalities

In this section we describe two mixed integer linear programming formulations for the MVTPP-ESC. The first one is a three-index one-commodity flow formulation. The second one introduces connectivity constraints to avoid subtours. Connectivity constraints are exponential in number. Since it is not computationally possible to introduce all such constraints in a MILP solver model, they are usually removed from the formulation and a separation algorithm solving a min-cut/max-flow problem is used to add only violated inequalities. On the contrary, in the three-index single-commodity flow formulation, the number of constraints, although large, remains polynomial thus avoiding the recourse to separation algorithms.

2.1 Three-index one-commodity flow formulation

To correctly introduce the commodity flow formulation we represent each edge $e \in E$ of our indirected graph as a pair of directed arcs (i, j) and (j, i), imposing that, for each pair of nodes, no more than one arc can be used, and define $A := \{(i, j) : i \in V, j \in V, i \neq j\}$ as the arcs set. For any 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 $i \in V \setminus S$. When a single node $i \in V$ is considered, we will write $\delta^+(\{i\})$ or $\delta^-(\{i\})$.

Let us introduce the following sets of binary and continuous variables:

$$x_{ij}^v := \begin{cases} 1 & \text{if arc } (i,j) \text{ is crossed by vehicle } v \\ 0 & \text{otherwise} \end{cases}, i \in V, j \in V, v \in F,$$

$$y_i^v := \begin{cases} 1 & \text{if supplier } i \text{ is visited by vehicle } v \\ 0 & \text{otherwise} \end{cases}, i \in M, v \in F,$$

$$w_k^v := \begin{cases} 1 & \text{if product } k \text{ is loaded on vehicle } v \\ 0 & \text{otherwise} \end{cases}, k \in K, v \in F,$$

$z_{ik}^v \geq 0$ quantity of product k purchased in supplier i by vehicle v, $k \in K, i \in M_k, v \in F$.

In commodity flow models, the connectivity requirement is imposed through a set of

continuous variables representing the flow of one or more "commodities" from the depot to the suppliers. In our model, a single commodity is sent from the depot in an amount equal to the total demand $\sum_{k\in K} d_k$ and is absorbed by each visited supplier i in an amount equal to the quantity purchased $\sum_{k\in K} z_{ik}^v$. We define as $f_{ij}^v \geq 0$, $(i,j) \in A$, $v \in F$, the flow variables representing the quantity on vehicle v when it leaves supplier i and arrives in j. The flow variables will define a number of paths stemming from the depot exactly equal to the number of optimal routes and each path will establish a sequence of suppliers that will be imposed to be part of an elementary tour starting and ending to the depot.

The MVTPP-ESC can be formulated as follows:

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

$$s.t. \qquad \sum_{v\in F}\sum_{j\in M} f_{0j}^v = \sum_{k\in K} d_k \tag{2}$$

$$\sum_{(i,j)\in\delta^+(\{i\})} f_{ij}^v - \sum_{(i,j)\in\delta^-(\{i\})} f_{ij}^v = -\sum_{k\in K} z_{ik}^v \quad i \in M, v \in F \tag{3}$$

$$f_{ij}^v \leq Qx_{ij}^v \quad (i,j) \in A, v \in F \tag{4}$$

$$x_{ij}^v + x_{ji}^v \leq 1 \quad i,j \in V, i \neq j, v \in F \tag{5}$$

$$\sum_{i\in M_k}\sum_{v\in F} z_{ik}^v = d_k \quad k \in K \tag{6}$$

$$\sum_{v\in F} z_{ik}^v \leq q_{ik} \quad k \in K, i \in M_k \tag{7}$$

$$z_{ik}^v \leq q_{ik}y_i^v \quad k \in K, i \in M_k, v \in F \tag{8}$$

$$w_k^v + w_g^v \leq 1 \quad (k,g) \in B, v \in F \tag{9}$$

$$w_k^v \leq \sum_{i\in M_k} z_{ik}^v \leq d_k w_k^v \quad k \in K, v \in F \tag{10}$$

$$x_{ij}^v \in \{0,1\} \quad (i,j) \in A, v \in F \tag{11}$$

$$y_i^v \in \{0,1\} \quad i \in V, v \in F \tag{12}$$

$$w_k^v \in \{0,1\} \quad k \in K, v \in F \tag{13}$$

$$f_{ij}^v \geq 0 \quad (i,j) \in A, v \in F \tag{14}$$

$$z_{ik}^v \geq 0 \quad k \in K, i \in M_k, v \in F \tag{15}$$

Objective function (1) establishes the minimization of traveling and purchasing costs. Constraint (2) states that the commodity flow leaving the depot is equal to the total demand. For each vehicle $v \in F$, the flow conservation constraints (3) establish that $\sum_{k\in K} z_{ik}^v$ units

of commodity are purchased from supplier i, $i \in M$. Constraints (4) imply that a flow for a given vehicle v can be sent along an arc (i, j) only if the arc has been traversed (i.e. only if $x_{ij}^v = 1$). Constraints (5) guarantee that each arc can be traversed in only one direction. Constraints (6) ensure that each product demand must be satisfied. Constraints (7) state that for each supplier the quantity loaded for a product k by all vehicles cannot exceed the product supply. Constraints (8) imply that if vehicle v loads a quantity of product k in supplier i then the supplier must be in its solution tour ($y_i^v = 1$). Inequalities (9) and (10) model the exclusionary side constraints. The first group of constraints state 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 ensure 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 is free to purchase the product in any supplier that sells it, but the total quantity cannot exceed value d_k. Finally, (11)-(15) are variables non-negativity and integrality conditions.

This formulation has the merit to be compact, and then can be easily implemented by means of a common MILP solver such as Cplex. Nevertheless, since the number of variables required is quite large and the continuous relaxation provided quite weak, reaching the optimal solution in a reasonable amount of time is often impossible in large size instances.

2.2 Connectivity constraints based formulation

A connectivity constraints based formulation can be obtained from (MVTPP-ESC$_{flow}$) by substituting inequalities (2), (3), (4), and (14) with the following ones:

$$\sum_{(i,j)\in\delta^+(\{h\})} x_{ij}^v = \sum_{(i,j)\in\delta^-(\{h\})} x_{ij}^v = y_h^v \quad h \in V, v \in F \tag{16}$$

$$\sum_{(i,j)\in\delta^+(S)} x_{ij}^v \geq y_h^v \quad S \subseteq M, h \in S, v \in F \tag{17}$$

$$\sum_{k\in K} \sum_{i\in M_k} z_{ik}^v \leq Q \quad v \in F. \tag{18}$$

We call the resulting model (MVTPP-ESC$_{conn}$). In constraints (16) we impose that if vehicle v visits a supplier h (i.e. $y_h^v = 1$), it also has to leave it. Connectivity inequalities (17) allow to eliminate subtours. Finally, constraints (18) impose that the quantity loaded on each vehicle does not exceed its capacity.

2.3 Valid Inequalities

In the following, we propose some classes of valid inequalities that will be used in the branch-and-cut approach to solve the (MVTPP-ESC$_{conn}$) formulation.

The first class of inequalities limits the value of binary variables associated with exclusionary constraints with the lower bound on the number of vehicles required to satisfy product demands. In particular, the total number of vehicles on which a product $k \in K$ can be loaded, i.e. $\sum_{v \in F} w_k^v$, cannot be lower than the rounded up ratio between the demand of k and the capacity of the vehicle:

$$\sum_{v \in F} w_k^v \geq \lceil d_k/Q \rceil \quad k \in K. \tag{19}$$

The second class of inequalities imposes that if a product k is purchased at a supplier i using vehicle v ($z_{ik}^v \geq 0$) then such a vehicle has to load the product k ($w_k^v = 1$):

$$z_{ik}^v \leq \min\{q_{ik}, d_k\} w_k^v, \quad i \in M_k, v \in F, k \in K. \tag{20}$$

A third class of inequalities establishes that if a product k is loaded on a vehicle v, at least one of the supplier offering that product have to be visited by that vehicle:

$$\sum_{i \in M_k} y_i^v \geq w_k^v \quad k \in K, v \in F. \tag{21}$$

Finally, the last class considers a generalization to the multi-vehicle case of SEC inequalities for TPP which involve z-variables (see Laporte et al. [17]):

$$\sum_{(i,j) \in \delta^+(S)} \sum_{v \in F} x_{ij}^v \geq \sum_{i \in S} \sum_{v \in F} z_{ik}^v / d_k \quad k \in K, S \subseteq M_k. \tag{22}$$

Each inequality (22) imposes that at least one arc must enter a subset $S \subseteq M_k$ whenever some amount of any product is purchased in at least one supplier of S. This type of inequalities can be disaggregated into $O(|F|)$ cuts as follows:

$$\sum_{(i,j) \in \delta^+(S)} x_{ij}^v \geq \sum_{i \in S} z_{ik}^v / d_k, \forall k \in K, S \subseteq M_k, v \in F. \tag{23}$$

.

3 The branch-and-cut approach

We propose a branch-and-cut framework to solve the (MVTPP-ESC$_{conn}$) formulation. Although the optimal integer solution provided by the two formulations is the same, (MVTPP-ESC$_{conn}$) contains less variables and its linear relaxation is stronger than that of (MVTPP-ESC$_{flow}$) providing a better lower bound for the problem. This means that (MVTPP-ESC$_{conn}$) is a better candidate for a branch-and-cut approach. The drawback is represented by constraints (17), that are exponential in the number of suppliers, forcing us to separate

them dynamically (i.e., during the branch-and-cut searching tree). Our implementation of the branch-and-cut also includes a preprocessing routine, the separation of some classes of valid inequalities and a starting integer initial solution obtained using the four-step heuristic described in Section 4.

3.1 Preprocessing

Preprocessing routine aims at determining some useful information from the input data. In particular, it checks for suppliers needed for the satisfaction of the demand, i.e. it computes the set $M^* := \left\{ i \in M : \exists k \in K : \sum_{j \in M \setminus \{i\}} q_{jk} < d_k \right\}$, and use it to add the following inequalities:

$$\sum_{v \in F} y_i^v \geq 1, \quad i \in M^*.$$

Moreover, it computes the set $K^* := \left\{ k \in K : \sum_{i \in M} q_{ik} = d_k \right\}$ corresponding to products without decision options to substitute some inequalities (7) with the following:

$$\sum_{v \in F} z_{ik}^v = q_{ik}, \quad i \in M, k \in K^*.$$

Finally, since products demands have to be satisfied exactly, we can replace coefficients q_{ik} with $min\{q_{ik}, d_k\}, i \in M, k \in K$.

3.2 Separation procedures

In this section we describe separation algorithm for valid inequalities (17) and (22). Given a fractional solution (x^*, y^*, z^*, w^*), these cuts can be separated exactly in polynomial time using ad hoc max-flow separation problems.

Separation procedure for inequalities (17). Given a vehicle v, let us define the capacitated graph $G_v = (V, A)$, obtained from G by imposing on each arc $(i, j) \in A$ a capacity equal to the value of x_{ij}^{v*}. Now, given a supplier i such that $y_i^{v*} \neq 0$, the most violated constraint (17) corresponds to a minimum-capacity cut $(S, V \setminus S)$ in G_v separating node 0 from i, and such that $i \in S$. This cut can be easily determined in $O(|M|^3)$ time by computing a maximum flow in G_v from 0 to i. Since a max-flow is solved for each $i \in M$ and each $v \in F$, the entire separation problem for inequalities (17) can be executed in $O(|M|^3|F|)$ time.

Separation procedure for inequalities (22). Since $\sum_{i \in M} \sum_{v \in F} z_{ik}^v = d_k$, inequalities (22) can be rewritten as follows:

$$\sum_{(i,j) \in \delta^+(S)} \sum_{v \in F} x_{ij}^v + \sum_{i \in M \setminus S} \sum_{v \in F} z_{ik}^v / d_k \geq 1 \quad \forall S \subseteq M, \forall k \in K.$$

Hence, given a product k, we can formulate a max-flow problem on a capacitated graph $G_k := (V', A')$ where $V' = V \cup \{t\}$ (consider t as a dummy node), and $A' = A \cup \{(i,t) : i \in M\}$. We set the capacity of each arc $(i,j) \in A$ equals to $\sum_{v \in F} x_{ij}^{v*}$, and the capacity of each new arc (i,t) equal to $\sum_{v \in F} z_{ik}^{v\,*}/d_k$. Now, let $(\bar{V}, V' \setminus \bar{V})$ be a minimum-capacity cut in G' separating 0 and t, with $t \in \bar{V}$. If the capacity is less than 1, then $S := \bar{V} \setminus \{t\}$ yields the most violated constraint (22). Since a max-flow, that can be executed in $O(|M|^3)$, is solved for each $k \in K$, the entire separation problem for inequalities (22) has a $O(|M|^3|K|)$ time complexity.

Finally, also inequalities (23) are exponential in number. However, we decide not to implement any separation procedure for these classes (see details in Section 5.2).

3.3 Breaking symmetry

In the model used for developing our branch-and-cut all the variables are indexed by vehicles (this is true for the flow formulation too). Although there exist two-index formulations for vehicle routing problems that ignore vehicles (when they are identical), in our case vehicles index is forced by the exclusionary constraints. This leads to a larger number of variables and, since we have an homogeneous fleet, to symmetry issues. In fact a feasible solution can be represented by every permutations of the vehicle indexes, that are $O(|F|!)$ in number. This means that each route in our feasible solution can be selected by a different vehicle without damaging the feasibility or the objective function value. Since symmetry could be penalizing in enumerative methods, we decide to globally break it including in the model the following inequalities:

$$y_0^v \leq y_0^{v-1}, \quad v \in F \setminus \{1\}. \tag{24}$$

Constraints (24) introduce an order in the use of the vehicles, allowing a vehicle to be used only if the preceding vehicle has already been used.

4 A four-step heuristic

We introduce a simple heuristic procedure based on a smart decomposition of the MVTPP-ESC. We consider four different decision steps, each one treated in a different way and depending on the decision taken in the previous one, as follows:

1. *The suppliers selection*: this step decides which suppliers will make part of a solution. In our algorithm, the supplier selection is tackled by an enumerative scheme limited through a Beam Search heuristic. The method identifies promising set of suppliers on the basis of the cost evaluation provided by the three following subproblems.

2. *The purchasing plan*: this step corresponds to the solution of a subproblem that decides the quantity for each product that has to be purchased at the suppliers, so that products demand is satisfied. The suppliers are those selected in step 1.

3. *The vehicles scheduling*: this subproblem decides, for each vehicle, which suppliers have to be visited, which quantity of each product has to be collected in each supplier, and consequently which products are loaded on which vehicle All these assignments have to be done guaranteeing the exclusionary side constraints. We implemented three greedy heuristics to solve this subproblem and an exact recovery method that minimizes the number of vehicles used.

4. *The vehicles routing*: in this step a route (a cycle starting and ending at the depot and visiting all chosen suppliers) is selected for each vehicle. Since we want to minimize traveling costs, a Traveling Salesman Problem (TSP) has to be solved for each vehicle and this is done heuristically.

4.1 The suppliers selection

Problem defined in step 1 is solved by means of a Beam Search (BS) strategy on a *balanced tree* in which each node is represented by a binary vector of size $|M|$ corresponding to a particular combination of selected (value set to 1) and non-selected (value set to zero) suppliers. More precisely, the *root node* of our searching tree has all the suppliers selected (all values to 1), and each child node has all selected supplier of the parent node minus one. However, since necessary suppliers could exist, only suppliers in $M \setminus M^*$ are eliminated from the solution during the search. We call l-st *level* (of the tree) the set of nodes that have the same number of non-selected suppliers equal to l. Figure 2 shows the structure of the tree for a sample problem with $|M| = 4$.

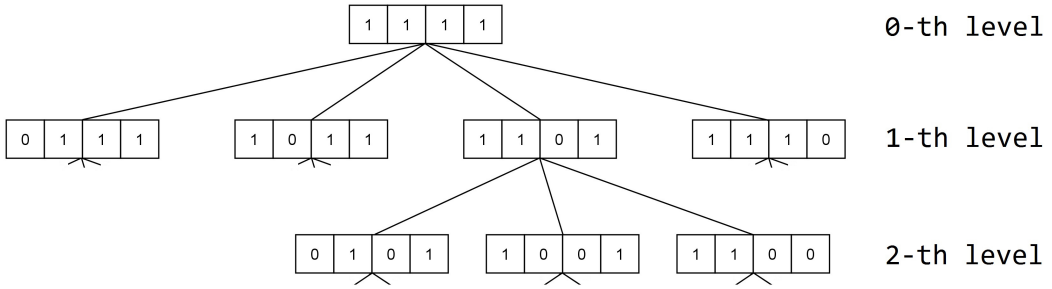


Figure 2: Supplier selection's exploration tree

The Beam Search is a heuristic algorithm that explores a graph or a tree by expanding

only a limited set of promising nodes (see e.g. Akyüz et al. [1] for a recent employment of a BS in a single and multi-commodity multi-facility problem). The main idea of our algorithm is indeed to evaluate heuristically the goodness of the nodes, and continue the search expanding only those most promising. This procedure is pseudocoded in Algorithm 1. In particular, for each *level* of the tree, the algorithm evaluates all nodes within the set $remainingNodes^{level}$. The $remainingNodes^0$ set is initialized with the root node, that contains all the suppliers. Each other $remainingNodes^{level}$ set is dynamically filled with all the children of a subset ($best^{level-1}$) containing the p most promising nodes in $remainingNodes^{level-1}$, where p is a given parameter. Algorithm 1 continues until a threshold time T_{MAX} is reached or no nodes remain to be evaluated.

Algorithm 1 Beam Search

1: $level \leftarrow 0$;
2: $remainingNodes^0 = \{rootNode\}$;
3: **while** ($remainingNodes^{level} \neq \emptyset$ **and** T_{MAX} is not reached) **do**
4: Evaluate all nodes in $remainingNodes^{level}$;
5: Choose the $best^{level}$, i.e. the p most promising nodes in $remainingNodes^{level}$;
6: Add to $remainingNodes^{level+1}$ all the children of each node in $best^{level}$;
7: $level \leftarrow level + 1$;
8: **end while**

The procedure used for evaluating a node is pseudocoded in Algorithm 2. Given a node, i.e. a subset of suppliers, its evaluation is done solving sequentially the other three subproblems. More precisely, we optimally solve the purchasing plan subproblem by means of an LP solver. If a feasible purchasing plan is found, this is completed by choosing a feasible vehicles scheduling and a feasible route for each vehicle, and assigning the resulting total cost to the current node. Otherwise, if the demand can not be satisfied by the selected suppliers, the current node is discarded from the tree. The rationale behind this is that an infeasible node can only generate an infeasible subtree, since each generated child has one supplier less than its parent.

4.2 The purchasing plan

Let $\bar{M}$ be the set of suppliers selected in a node of the tree. To optimize the purchasing plan we formulate the following subproblem:

$$min \sum_{k \in K} \sum_{i \in \bar{M}} p_{ik} z_{ik} \tag{25}$$

Algorithm 2 Evaluating a node

1: Optimize the *purchasing plan* subproblem;
2: **if** the *purchasing plan* is infeasible **then**
3: Delete the node from the tree;
4: **else**
5: Find a feasible *vehicles scheduling*;
6: Find a feasible *vehicles routing*;
7: Assign to the node the value of the resulting total cost;
8: **end if**

$$s.t. \quad \sum_{i \in \bar{M}} z_{ik} = d_k \quad k \in K \tag{26}$$

$$0 \leq z_{ik} \leq q_{ik} \quad k \in K, i \in \bar{M} \tag{27}$$

where $z_{ik} := \sum_{v \in F} z_{ik}^v$ are continuous variables. Since this problem can be decomposed into $|K|$ Transportation Problems it is easy-to-solve. We decide to solve it optimally through an LP solver and we call $\bar{z}$ the resulting optimal solution.

4.3 The vehicles scheduling

We propose three procedures for solving the *vehicles scheduling* subproblem, each one focused on a different optimization aspect. In particular, $\mathtt{H}_{Sup}$, $\mathtt{H}_{Esc}$, and $\mathtt{H}_{Prod}$ are greedy heuristics where the assignment of products to vehicles and that of vehicles to the suppliers are done supplier by supplier, supplier by supplier but privileging incompatible products, and product by product, respectively.

Given an initial set of suppliers, $\mathtt{H}_{Sup}$ selects randomly a supplier to serve. Then, chosen a vehicle within the fleet, it assigns that vehicle to the supplier and loads as much as possible of the available products provided that: a) quantity loaded for each product is bounded by the optimal *purchasing plan*; b) exclusionary constraints are satisfied; c) the capacity of the vehicle is not overflowed. The priority on the use of vehicles is controlled by their current distances from the considered supplier, i.e. $\mathtt{H}_{Sup}$ uses first the vehicle that is located at the closest supplier. All vehicles are initially placed at the depot. If a vehicle is not sufficient to completely serve a supplier, the second vehicle in order of priority is chosen, and so on. This procedure is repeated for each supplier until they are all served.

The method $\mathtt{H}_{Esc}$ works exactly as the previous one, but it separately considers incompatible and free products. More precisely, the algorithm runs twice the $\mathtt{H}_{Sup}$ procedure. The first time $\mathtt{H}_{Sup}$ will consider only incompatible products, then the vehicles are brought back to the depot. The second time $\mathtt{H}_{Sup}$ is executed considering only the remaining free products.

Greedy $\mathtt{H}_{Prod}$ tries to complete the purchasing of each incompatible product first, irrespective of the suppliers that offer it, and to assign it to the minimum number of vehicles. More precisely, it chooses randomly an incompatible product. Then, selected randomly a vehicle within the fleet, it assigns that vehicle to the product and loads on it all the possible quantity provided by the suppliers. If a vehicle is not sufficient (capacity constraint) or not allowed (exclusionary constraint) to completely load a product, another vehicle is chosen, and so on. This procedure is repeated for each incompatible product until they are all purchased. Finally, for each supplier, the remaining free products are purchased filling up the vehicles that have been already associated to that supplier.

Since the proposed greedy heuristics can sporadically terminate without finding a feasible solution, we also implement a recovery procedure (`Veh`) consisting in the optimal resolution of the following model focused on the minimization of the number of vehicles used:

$$min \sum_{v \in F} r^v \tag{28}$$

$$s.t. \quad \sum_{v \in F} z_{ik}^v = \bar{z}_{ik} \quad k \in K, i \in \bar{M} \tag{29}$$

$$z_{ik}^v \leq q_{ik} y_i^v \quad k \in K, i \in \bar{M}, v \in F \tag{30}$$

$$\sum_{v \in F} y_i^v \geq 1 \quad i \in \bar{M} \tag{31}$$

$$w_k^v + w_g^v \leq 1 \quad (k,g) \in B, v \in F \tag{32}$$

$$w_k^v \leq \sum_{i \in \bar{M}} z_{ik}^v \leq d_k w_k^v \quad k \in K, v \in F \tag{33}$$

$$\sum_{i \in \bar{M}} y_i^v \leq |M| r^v \quad v \in F \tag{34}$$

$$y_i^v \in \{0,1\} \quad i \in \bar{M}, v \in F \tag{35}$$

$$w_k^v \in \{0,1\} \quad k \in K, v \in F \tag{36}$$

$$r^v \in \{0,1\} \quad v \in F \tag{37}$$

$$z_{ik}^v \geq 0 \quad k \in K, i \in \bar{M}, v \in F \tag{38}$$

Variables r^v are binary and take value 1 if vehicle v is used, and 0 otherwise. Beside the already explained inequalities, constraints (29) state that the quantity loaded for each product in each supplier has to be equal to the optimal *purchasing plan* ($\bar{z}$), constraints (31) force vehicles to visit suppliers currently selected (set $\bar{M}$) at least once, and each constraints (34) sets $r^v = 1$ if vehicle v visits at least one supplier. As already said, the objective function (28) minimizes the number of vehicles used. The optimal solution of this subproblem is obtained by means of a MILP solver.

4.4 The vehicles routing

Once decided which suppliers have to be visited by a vehicle, the routing subproblem aims at finding a visiting tour for each vehicle so to minimize the global traveling cost. As states before, this corresponds to solving a TSP for each vehicle. In our implementation this is done using the Lin-Kernighan-Helsgaun (LKH) heuristic (Helsgaun [15]), that is a recent variant of one of the most successful methods for generating optimal or near-optimal solutions for the symmetric TSP.

5 Computational experiments

This section is devoted to present the setting and the results of our computational experiments. We first aim to compare our exact approach, as described in Section 3 with the plain resolution of the (MVTPP-ESC$_{flow}$) model and then to evaluate efficiency and effectiveness of our four-step heuristic. We use Cplex 12.3 to solve optimally subproblems and as a general branch-and-cut framework, implementing and incorporating our separation procedures with the C++ Concert Technology's callbacks. The algorithms have been coded in C/C++ and tested on an *Intel Core i7* (quad core) computer, with 8 GB RAM and running *Windows 7* 64-bit operating system.

5.1 Instances

Since the MVTPP-ESC has never been studied before, no benchmark instances are available in the literature. We decided to build our problems starting from benchmark instances available for the TPP in Laporte et al. [17]. More precisely, the suppliers integer coordinates are randomly generated in a $[0, 1000] \times [0, 1000]$ square according to a uniform distribution and routing costs are computed as Euclidean distances. Each product k is associated with $|M_k|$ randomly selected suppliers, where $|M_k|$ is randomly generated in the interval $[1, |K|]$. Product prices p_{ik} are selected in interval [1,10] according to a discrete uniform distribution. For each product k and each supplier i, q_{ik} is randomly generated in [1,15], and $d_k = \lambda \max_{i \in M_k} q_{ik} + (1 - \lambda) \sum_{i \in M_k} q_{ik}$ with $\lambda \in [0, 1]$. The bigger the λ value, the lower the number of suppliers selected in an optimal solution.

To generate exclusionary sets, we firstly fix the percentage of free products $f\%$, i.e. the products that can be transported together. Then, for each incompatible product, we randomly choose the number and the set of products with which there is an incompatibility and we construct the exclusionary pairs. The structure of the instance can be better controlled this way.

Finally, we need to determine the number u of available vehicles in order to guarantee

that at least a feasible assignment of products to the vehicles exists. If we do not consider the vehicles capacity, this assignment can be found by solving the well-known Minimum Vertex Coloring Problem (MVCP), which tries to color all the nodes of a given a graph $G^c = (V^c, E^c)$ in such a way that two adjacent nodes do not have the same color, minimizing the number of colors used. In our case, $V^c = K$ and $E^c = B$ (i.e. each node corresponds to a product and two products are adjacent in G^c if and only if they are incompatible), and each color used corresponds to a different vehicle. Hence, the optimal solution value of MVCP (i.e. the minimum number of colors needed) represents a lower bound for the number of vehicles required. Not only, but the optimal solution of the MVCP establishes which product is assigned to each vehicle. To be sure that the number of vehicles is enough to satisfy demand, that value has to be adjusted on the basis of the vehicle capacity Q. Then, for each vehicle, we calculate the sum of demands of assigned products and compare it with Q: if that vehicle can transport all the assigned products, only a vehicle is needed, otherwise the number of necessary vehicles is obtained rounding up the ratio of the total demand over Q. The rationale for solving an $\mathcal{NP}$-hard problem like the MVCP, instead of assigning a large value to u, is to keep the number of vehicles as small as possible, given that the MVTPP-ESC formulations size highly depend on u.

For our computational experiments we consider instances with a number of suppliers $m = \{20, 35, 50\}$, a number of products $n = \{50, 100\}$, and two values for λ parameter, i.e. $\lambda = \{0.1, 0.8\}$. We also consider two percentage values of free products $f\%$, 80% and 90%. This means that the more complex instances have 20% of cross-incompatibilities between products. Moreover, we use two different capacities Q of vehicles, i.e. $\{1500, 3000\}$ for instances with 90% of free products, and $\{2000, 4000\}$ for those with 80% of free products. Finally, 3 different instances ($\# := \{1, 2, 3\}$) for each combination of $\{|V|, |K|, \lambda, Q, f\%\}$ have been generated, this means 144 instances all together. Instances can be downloaded at the web page *http://www.ing.unibs.it/∼orgroup/instances.html*.

5.2 Implementation details

The streamlined version of our branch-and-cut has several implementing details, derived from some preliminary tuning tests and specified in the following:

- valid inequalities (19), (20) and (21), that are polynomial in number, are introduced directly in the model, and not separated dynamically. Since in our instances the number of linear relaxation solved is huge, it seems to be less time consuming to work with an initial slightly bigger model than to spend too much CPU time checking for these cuts at each relaxation.

- inequalities (22) come out to be not so effective to strengthen the model, while their

separation procedure is very time consuming. Hence, we decide to separate them only at the root node. Also inequalities (23) are not separated for the same reasons;

- in order to avoid out-of-memory problems caused by the massive adding of inequalities (17) during the branch-and-cut tree, we decide to separate them only at the root node, and to add them in a *lazy* way during the rest of the tree. Separating a cut in a *lazy* way means that the procedure is activated only when an integer solution is found, and not for each linear relaxation solution;

- as stated in Section 3.2, for separating valid inequalities (17) and (22) we need to solve max-flow problems. Although different polynomial-time algorithms exist in the literature (see for instance, Goldberg and Tarjan [12]) we prefer to solve it through the optimal solution of its dual problem, the min-cut problem, by means of an LP solver. While using a solver does not affect too much the computational time of a single max-flow problem on respect to an ad hoc procedure, the real time saving occurs when a great number of very similar problems have to be solved serially, as in our case. Our method allows us to solve a new separation by re-optimizing on a min-cut model after only changing very few coefficients.

5.3 Computational results and analysis

We first analyze the three variants of the proposed heuristic obtained associating the Beam Search to one of the greedy methods proposed for the vehicle scheduling subproblem. We named them BS+H$_{Sup}$, BS+H$_{Esc}$, and BS+H$_{Prod}$. We have also decided to test, as an additional method, the heuristic obtained associating the Beam Search to the optimal solution of the model (28)-(38), and we named it BS+Veh. The best performing variant will be used to provide an input solution to our branch-and-cut approach. In these procedures we have set the number p of best promising nodes to expand for each level equal to 4, and the threshold time T_{MAX} equal to 120 seconds.

Table 1 reports the results for the four methods on a subset of 24 instances with $f\% = 90$, uniquely identified by the combination of parameters $\{|V|, |K|, \lambda, Q\}$. For each method the objective function (*obj*) of the best solution found, the percentage error ($\Delta\% = [z^H - z^*] \times 100/z^*$, where z^H is the solution value of the corresponding method and z^* is the value of the best solution found), and the computational time t in seconds are specified. For each instance, when the objective function value found by a method is the best one among the four variants, the corresponding figure is highlighted in bold font in column *obj*. It is evident how, on average, BS+H$_{Prod}$ outperforms all the other methods, reaching the best solution in 22 out of 24 instances and requiring the lowest average time. BS+H$_{Sup}$ and BS+H$_{Esc}$ have a slightly worse performance than BS+H$_{Prod}$ achieving an average percentage error equal to 0.48

and 0.41, respectively, whereas BS+Veh is strongly overcome (in computational time and in solution quality) by the other heuristics. Notwithstanding in BS+Veh the vehicles scheduling is done by optimally solving a model, the minimization of the number of vehicles used as objective function probably leads to solutions with higher traveling costs, and consequently to a worse evaluation of nodes in the BS tree. Finally, BS+H$_{Prod}$ has been chosen as starting heuristic for the branch-and-cut. When it leads to an infeasible solution, the BS+Veh is used as recovery method.

Table 1: Comparison between variants of four-step heuristic

				BS+H$_{Sup}$			BS+H$_{Esc}$			BS+H$_{Prod}$			BS+Veh		
$\|V\|$	m	λ	Q	obj	Δ%	t	obj	Δ%	t	obj	Δ%	t	obj	Δ%	t
20	50	10	1500	871726	0.46	0	868318	0.06	0	**867759**	0.00	0	877429	1.11	0
20	50	10	3000	868300	0.11	0	868401	0.12	0	**867366**	0.00	0	873369	0.69	0
20	50	80	1500	188409	0.47	1	188409	0.47	1	**187535**	0.00	1	189948	1.29	9
20	50	80	3000	188409	0.47	1	188409	0.47	1	**187535**	0.00	1	189948	1.29	10
20	100	10	1500	1728519	0.52	0	1721982	0.14	0	**1719508**	0.00	0	1738362	1.10	1
20	100	10	3000	1723312	0.23	0	1721982	0.15	0	**1719357**	0.00	0	1730958	0.67	0
20	100	80	1500	312469	0.21	0	313362	0.50	0	**311800**	0.00	0	315675	1.24	10
20	100	80	3000	313362	0.41	0	313362	0.41	0	**312088**	0.00	0	316648	1.46	8
35	50	10	1500	**1365524**	0.00	0	1366326	0.06	0	1366245	0.05	0	1372780	0.53	0
35	50	10	3000	1361314	0.08	0	1360422	0.01	0	**1360223**	0.00	0	1364243	0.30	0
35	50	80	1500	185652	1.06	15	185660	1.06	16	**183710**	0.00	13	189150	2.96	120
35	50	80	3000	185182	0.80	17	185182	0.80	16	**183710**	0.00	13	186808	1.69	120
35	100	10	1500	2990262	0.11	0	2988226	0.05	0	**2986833**	0.00	0	3008446	0.72	2
35	100	10	3000	2981376	0.12	0	2978939	0.04	0	**2977859**	0.00	0	2990177	0.41	1
35	100	80	1500	375947	1.02	13	374982	0.77	15	**372133**	0.00	11	386585	3.88	120
35	100	80	3000	374768	0.73	14	374756	0.73	14	**372040**	0.00	11	380716	2.33	121
50	50	10	1500	1746209	0.48	1	1740642	0.16	1	**1737798**	0.00	1	1756933	1.10	12
50	50	10	3000	1738157	0.28	1	1735259	0.12	1	**1733246**	0.00	1	1747969	0.85	8
50	50	80	1500	202310	0.68	67	202133	0.59	62	**200948**	0.00	55	207126	3.07	120
50	50	80	3000	202070	0.56	64	202070	0.56	66	**200948**	0.00	54	203182	1.11	120
50	100	10	1500	3929091	0.24	0	3924293	0.11	0	**3919860**	0.00	0	3962182	1.08	13
50	100	10	3000	3919134	0.30	0	3911863	0.11	0	**3907498**	0.00	0	3936555	0.74	10
50	100	80	1500	406571	2.23	74	404253	1.65	81	**397710**	0.00	61	414886	4.32	121
50	100	80	3000	**394855**	0.00	67	397720	0.73	73	397792	0.74	56	407053	3.09	121
					0.48	14		0.41	15		0.03	12		1.54	44

Table 4-6 show the results obtained by our branch-and-cut approach (B&C) and by the plain resolution of (MVTPP-ESC$_{flow}$) formulation with Cplex 12.3 (CplexFlow) for instances with $|V| = 20, 30$, and 50, respectively. We set a threshold time for the two exact procedures equals to 2 hours, i.e. 7200 seconds. In each of these tables the first five columns $(f\%, |K|, \lambda, Q, \#)$ uniquely identify the solved instance. The consecutive two columns refer to the BS+H$_{Prod}$ heuristic and represent its percentage error (Δ%) with respect to the best solution found by B&C and its computational time t in seconds. Concerning B&C, the column headings have the following meaning: *obj* is the value of the best solution found (possibly the optimal one), *gap*% is the percentage gap of this solution from the best lower bound found in the branch-and-cut tree, t is the CPU time in seconds, and *ttb* (time to best) is

the CPU time needed to find the best feasible solution. Concerning `CplexFlow`, beyond the *gap*%, t and *ttb* columns, we report the percentage error (Δ%) of this method with respect to the best solution found by `B&C`. In columns *gap*%, the entries are highlighted in bold font, when the optimal solution has been found, i.e. *gap*% = 0.00.

`CplexFlow` method seems to have a very poor performance in almost all tested instances. In 44 out of 144 instances, it has even not been able to find a feasible solution within 2 hours of computational time (6, 17 and 21 instances with $|V| = 20, 35$ and 50, respectively). In the remaining instances, it has been able to find the optimal solution only twice, and the average gaps are quite high, i.e. 1.40%, 3.25% and 4.66% for $|V| = 20, 35$ and 50 instances, respectively. These results strongly confirm preliminary findings in Manerba and Mansini [18].

Let's now look at `B&C` performance. The method has been able to achieve 30 optimal solutions, the most part of which in instances with $|V| = 20$. Moreover, the average optimality gaps (equal to 0.63%, 1.44% and 1.62% for $|V| = 20, 35$ and 50 instances, respectively) are more than halved in the first two cases with respect to `CplexFlow` gaps, while the last gap is almost a third. Apart from gaps, the quality of the solution is, on average, better as shows in column Δ% of `CplexFlow`: when available, the average percentage error with respect to our method is quite small for $|V| = 20$ (i.e. 0.15%), but then it increases to 0.89 and 1.63 for $|V| = 35$ and 50, respectively. `CplexFlow` outperforms `B&C` providing a better solution only in 3 instances (see the negative values in Δ% column), but the percentage errors are really negligible. Finally, the results of our heuristic `BS+H`$_{Prod}$ also deserves a particular attention. Columns Δ% of `BS+H`$_{Prod}$ in Tables 4-6 show that the average errors with respect to the best solution found by `B&C` are very small (0.53%, 0.30% and 0.36% for $|V| = 20, 35$ and 50 instances, respectively). For 28 instances the heuristic method even terminates with the same solution found by the `B&C`. Since computational times of `BS+H`$_{Prod}$ can not exceed 120 seconds, our four-step heuristic represents a valid solving procedure.

For sake of completeness, in Tables 2 and 3 we conclude the analysis showing the most relevant information on our branch-and-cut method. In particular, Table 2 summarizes the `B&C` improvements with respect to `CplexFlow` results. For each combination of $\{|V|, |K|, \lambda, f\%, Q\}$, the table reports the total number of times, out of 3, where `B&C` reaches or outperforms `CplexFlow` solutions, whereas into brackets the number of strict improvements is shown. It can be clearly seen how our method reaches or outperform Cplex 12.3 in the quasi-totality of the instances, and is strictly the best method in about 83% of the cases. Finally, Table 3 reports the average number (on the 3 random generated instances) of violated valid inequalities (17) and (22) added, and the average number of nodes (#*nod*) visited in the branch-and-cut tree.

Table 2: B&C improvements

			$f\% = 90$		$f\% = 80$		
$\|V\|$	$\|K\|$	λ	$Q = 1500$	$Q = 3000$	$Q = 2000$	$Q = 4000$	Tot. impr
20	50	10	3(2)	3(1)	3(2)	3(1)	12(6)
	50	80	3(0)	3(0)	3(1)	3(1)	12(2)
	100	10	3(3)	3(3)	3(3)	3(3)	12(12)
	100	80	3(2)	3(2)	3(3)	3(3)	12(10)
35	50	10	2(2)	3(2)	3(3)	3(3)	11(10)
	50	80	2(2)	3(1)	2(2)	3(3)	10(8)
	100	10	3(3)	3(3)	3(3)	3(3)	12(12)
	100	80	3(3)	3(3)	3(3)	3(3)	12(12)
50	50	10	3(3)	3(3)	3(3)	3(3)	12(12)
	50	80	3(3)	3(3)	3(3)	3(3)	12(12)
	100	10	3(3)	3(3)	3(3)	3(3)	12(12)
	100	80	3(3)	3(3)	3(3)	3(3)	12(12)
			34(29)	36(27)	35(32)	36(32)	141(120)

Table 3: B&C details

$\|V\|$	$\|K\|$	λ	$f\% = 90, Q = 1500$			$f\% = 90, Q = 3000$			$f\% = 80, Q = 2000$			$f\% = 80, Q = 4000$		
			(17)	(22)	#nod	(17)	(22)	#nod	(17)	(22)	#nod	(17)	(22)	#nod
20	50	10	734	95	182501	743	122	17409	486	140	122893	897	109	179559
	50	80	632	127	467126	704	138	36779	366	108	2382	860	234	34273
	100	10	733	122	18606	743	68	113082	815	114	1375850	1088	257	27085
	100	80	1401	225	302623	1181	172	15945	1314	237	59365	2367	163	87604
35	50	10	1261	203	419409	3499	202	199613	1834	238	51420	5827	185	70351
	50	80	1761	184	117491	2094	164	294044	2396	170	52885	4860	173	73371
	100	10	2466	634	26	3619	332	38662	840	257	1035	11068	343	21230
	100	80	981	333	12530	2404	224	5228	2558	256	18597	2061	184	7396
50	50	10	2022	375	42028	3683	194	27922	6705	321	20539	5463	215	28331
	50	80	9563	195	42568	3052	382	44	8470	210	46605	2379	391	60
	100	10	8296	170	80657	2255	410	16930	12135	194	34555	2155	350	17900
	100	80	5255	445	5	4714	403	35018	2747	362	5	12501	308	11057

6 Conclusions

In this paper we study a new variant of the multi-vehicle traveling purchaser problem characterized by the presence of incompatible products that cannot be loaded on the same vehicle. The problem is extremely challenging and, to the best of our knowldge, it has never been studied up to now. A branch-and-cut algorithm is proposed, together with ad hoc classes of valid inequalities. The method provides on a large set of instances results that outperform, both in quality and in efficiency, those obtained by Cplex 12.3 when used as MILP solver to tackle a three-index single commodity flow formulation of the problem.

We also introduce a four-step heuristic procedure to find an initial feasible solution. The method itself comes out to be quite effective and extremely efficient, yielding a valuable solving method for the most difficult instances.

Finally, looking at the good results obtained by our four-step heuristic, we believe that its basic idea of tackling separated subproblems could be further studied, representing a good

starting point to develop a new stand-alone matheuristic procedure for the MVTPP-ESC.

References

[1] Akyüz, M. H., Öncan, T., Altinel, K. I., 2013. Beam search heuristics for the single and multi-commodity capacitated multi-facility weber problems. Computers & Operations Research 40 (12), 3056–3068.

[2] Angelelli, E., Mansini, R., Vindigni, M., 2009. Exploring greedy criteria for the dynamic traveling purchaser problem. Central European Journal of Operations Research 17, 141–158.

[3] Angelelli, E., Mansini, R., Vindigni, M., 2011. Look-ahead heuristics for the dynamic traveling purchaser problem. Computers & Operations Research 38, 1867–1876.

[4] Angelelli, E., Mansini, R., Vindigni, M., 2012. The stochastic and dynamic traveling purchaser problem. Working paper.

[5] Beraldi, P., Bruni, M. E., Manerba, D., Mansini, R., 2013. The traveling purchaser problem under uncertainty. (submitted).

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

[7] Burstall, R., 1966. A heuristic method for a job-scheduling problem. Operation Research Quart. 17 (3), 291–304.

[8] Cao, B., Uebe, G., 1995. Solving transportation problems with nonlinear side constraints with tabu search. Computers & Operations Research 22 (6), 593 – 603.

[9] 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.

[10] Dror, M., Trudeau, P., 1989. Saving by split delivery routing. Transportation Science 23, 141–145.

[11] Fallahi, A., Prins, C., Calvo, R. W., 2008. A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. Computers & Operations Research 35, 1725–1741.

[12] Goldberg, A., Tarjan, R., 1988. A new approach to the maximum-flow problem. Journal of the ACM 35, 921–940.

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

[14] Gouveia, L., Paias, A., Voss, S., 2011. Models for a traveling purchaser problem with additional side-constraints. Computers & Operations Research 38, 550–558.

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

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

[17] 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.

[18] Manerba, M., Mansini, M., 2013. Multi-vehicle traveling purchaser problem with exclusionary side constraints. In: Proceedings of VeRoLog2013 Conference. p. 63.

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

[20] Mendoza, J. E., Castanier, B., Guret, C., Medaglia, A. L., Velasco, N., 2011. Constructive heuristics for the multicompartment vehicle routing problem with stochastic demands. Transportation Science 45 (3), 346–363.

[21] Muyldermans, L., Pang, G., 2010. On the benefits of co-collection: Experiments with a multi-compartment vehicle routing algorithm. European Journal of Operational Research 206 (1), 93 – 103.

[22] Ramesh, T., 1981. Traveling purchaser problem. Opsearch 18, 78–91.

[23] Riera-Ledesma, J., Salazar-Gonzàlez, 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.

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

Appendix: Tables

Table 4: `B&C` vs `CplexFlow` on $|V| = 20$ instances

					`BS+H`$_{Sup}$		`B&C`				`CplexFlow`			
f%	$\|K\|$	λ	Q	#	Δ%	t	obj	gap%	t	ttb	Δ%	gap%	t	ttb
90	50	10	1500	1	0.07	0	867173	0.48	7200	2087	0.02	1.14	7200	6931
	50	10	1500	2	0.08	0	908238	0.29	7200	858	0.00	0.65	7200	7068
	50	10	1500	3	0.16	0	757537	**0.00**	236	105	0.00	0.22	7200	6729
	50	10	3000	1	0.02	0	867173	0.02	7200	1132	0.00	0.87	7200	7041
	50	10	3000	2	0.06	0	908238	**0.00**	1874	754	0.05	0.85	7200	7147
	50	10	3000	3	0.06	0	757537	**0.00**	49	36	0.00	0.22	7200	6882
	50	80	1500	1	0.65	1	186330	**0.00**	10	10	0.00	0.08	7200	121
	50	80	1500	2	0.38	1	162371	**0.00**	249	137	0.00	1.03	7200	6525
	50	80	1500	3	0.68	0	137517	**0.00**	20	15	0.00	0.15	7200	143
	50	80	3000	1	0.65	1	186330	**0.00**	10	10	0.00	**0.00**	1535	72
	50	80	3000	2	0.38	1	162371	**0.00**	209	70	0.00	0.94	7200	6541
	50	80	3000	3	0.68	0	137517	**0.00**	21	17	0.00	**0.00**	3693	104
	100	10	1500	1	0.01	0	1719310	0.84	7200	7071	-	-	7200	-
	100	10	1500	2	0.20	0	1742201	0.41	7200	7200	-	-	7200	-
	100	10	1500	3	0.17	0	1787437	0.58	7200	7200	0.09	0.68	7200	3240
	100	10	3000	1	0.01	0	1719229	0.53	7200	7200	0.27	1.12	7200	7200
	100	10	3000	2	0.20	0	1741900	0.30	7200	6843	0.01	0.66	7200	6918
	100	10	3000	3	0.04	0	1787286	0.37	7200	3839	0.50	1.12	7200	1270
	100	80	1500	1	1.15	0	308263	**0.00**	6713	558	0.01	1.41	7200	7200
	100	80	1500	2	0.74	1	313638	1.51	7200	7098	0.16	2.23	7200	7172
	100	80	1500	3	0.24	1	332741	**0.00**	2198	860	0.00	0.38	7200	6508
	100	80	3000	1	1.24	0	308263	**0.00**	3543	347	0.12	1.55	7200	6958
	100	80	3000	2	2.42	17	313623	0.13	7200	2106	0.10	1.22	7200	6830
	100	80	3000	3	0.10	1	332741	**0.00**	843	129	0.00	0.52	7200	6574
80	50	10	2000	1	0.38	0	866853	**0.00**	846	785	0.02	0.47	7200	7139
	50	10	2000	2	0.44	0	905475	**0.00**	1868	179	0.00	0.05	7200	6904
	50	10	2000	3	0.44	0	764166	**0.00**	389	110	0.01	0.65	7200	7140
	50	10	4000	1	0.11	0	866853	**0.00**	31	29	0.00	0.25	7200	6550
	50	10	4000	2	0.08	0	905475	**0.00**	45	28	0.00	0.02	7200	297
	50	10	4000	3	0.05	0	764171	**0.00**	31	27	0.03	0.18	7201	6532
	50	80	2000	1	0.05	2	191051	**0.00**	111	68	0.09	1.75	7200	6756
	50	80	2000	2	0.59	2	163769	**0.00**	98	56	0.00	0.88	7200	6500
	50	80	2000	3	0.84	1	140682	0.05	7200	161	0.00	3.17	7200	7016
	50	80	4000	1	0.05	2	191051	**0.00**	185	68	0.01	1.56	7200	6529
	50	80	4000	2	0.59	2	163769	**0.00**	123	59	0.00	1.01	7200	6580
	50	80	4000	3	0.84	1	140682	**0.00**	3122	688	0.00	2.44	7200	6805
	100	10	2000	1	0.32	0	1724280	1.09	7200	7156	-	-	7200	-
	100	10	2000	2	0.37	0	1750482	1.09	7200	7200	-	-	7200	-
	100	10	2000	3	0.01	0	1795719	1.05	7200	7200	-	-	7200	-
	100	10	4000	1	0.10	0	1724253	0.98	7200	6623	0.27	1.43	7200	3712
	100	10	4000	2	0.25	0	1749743	0.98	7200	6958	-	-	7200	-
	100	10	4000	3	0.23	0	1791756	0.67	7200	5085	0.37	1.23	7200	7200
	100	80	2000	1	0.44	0	316911	3.67	7200	7200	2.13	6.55	7200	7200
	100	80	2000	2	2.71	34	319140	2.63	7200	7200	0.31	3.71	7200	7134
	100	80	2000	3	1.71	44	339506	3.29	7200	7200	0.33	3.95	7200	7200
	100	80	4000	1	0.45	0	316887	3.78	7200	7200	0.72	5.08	7200	6539
	100	80	4000	2	2.45	33	319132	2.51	7200	7193	0.27	3.67	7200	7200
	100	80	4000	3	1.73	43	339431	3.08	7200	7100	0.25	3.84	7200	7200
					0.53	4		0.63	4076	2943	0.15	1.40	7009	5793

Table 5: B&C vs CplexFlow on $|V| = 35$ instances

					BS+H$_{Sup}$		B&C				CplexFlow			
f%	$\|K\|$	λ	Q	#	Δ%	t	obj	gap%	t	ttb	Δ%	gap%	t	ttb
90	50	10	1500	1	0.37	0	1361244	0.40	7200	7200	0.12	0.52	7200	7200
	50	10	1500	2	0.04	0	1437840	0.55	7200	7200	-0.04	0.50	7200	7200
	50	10	1500	3	0.31	0	1209113	0.42	7200	7200	0.16	0.64	7200	7200
	50	10	3000	1	0.08	0	1359082	**0.00**	100	43	0.01	0.04	7200	2991
	50	10	3000	2	0.10	0	1432005	**0.00**	707	630	0.00	0.04	7200	7002
	50	10	3000	3	0.03	0	1206324	**0.00**	139	135	0.00	0.07	7200	6838
	50	80	1500	1	0.37	12	183041	0.16	7200	6583	0.24	1.71	7200	7195
	50	80	1500	2	0.64	14	173777	0.65	7200	6847	-0.02	2.05	7200	7200
	50	80	1500	3	0.47	9	177365	0.10	7200	5604	0.03	2.56	7200	7200
	50	80	3000	1	0.37	12	183041	**0.00**	6753	745	0.00	0.89	7200	7100
	50	80	3000	2	0.71	13	173657	**0.00**	305	152	0.00	1.41	7200	6955
	50	80	3000	3	0.47	9	177365	**0.00**	1820	639	0.28	2.04	7200	7199
	100	10	1500	1	0.00	0	2986833	0.80	7200	209	-	-	7200	-
	100	10	1500	2	0.00	0	2917799	0.68	7200	456	-	-	7200	-
	100	10	1500	3	0.00	0	2993262	0.76	7200	524	-	-	7200	-
	100	10	3000	1	0.01	0	2977581	0.51	7200	7200	-	-	7200	-
	100	10	3000	2	0.01	0	2910222	0.43	7200	7200	-	-	7200	-
	100	10	3000	3	0.00	0	2986465	0.55	7200	158	-	-	7200	-
	100	80	1500	1	0.91	10	368794	1.86	7200	7016	3.66	5.74	7200	972
	100	80	1500	2	0.23	12	353152	1.97	7200	7200	2.70	4.90	7200	702
	100	80	1500	3	0.10	11	368082	3.40	7200	7200	1.79	5.41	7200	2803
	100	80	3000	1	0.93	10	368611	1.60	7200	7200	1.88	4.03	7200	752
	100	80	3000	2	0.23	12	352827	1.82	7200	7134	1.40	3.51	7200	7200
	100	80	3000	3	0.01	11	368179	3.28	7200	7088	-	-	7200	-
80	50	10	2000	1	0.04	0	1365599	0.52	7200	7145	-	-	7200	-
	50	10	2000	2	0.06	0	1441876	0.45	7200	6929	0.23	1.09	7200	7195
	50	10	2000	3	0.06	0	1212349	0.50	7200	7200	-	-	7200	-
	50	10	4000	1	0.04	0	1365504	0.32	7200	7061	0.08	0.83	7200	7148
	50	10	4000	2	0.06	0	1441768	0.33	7200	7200	0.50	1.37	7200	7200
	50	10	4000	3	0.07	0	1212217	0.41	7200	7200	0.05	0.83	7200	7200
	50	80	2000	1	0.85	15	189733	2.82	7200	6998	0.56	5.41	7200	7200
	50	80	2000	2	0.83	15	180704	4.46	7200	7144	-0.03	4.93	7200	7200
	50	80	2000	3	1.12	8	183183	3.79	7200	7200	0.85	5.99	7200	7200
	50	80	4000	1	0.82	15	189789	2.88	7200	7007	0.30	4.95	7200	7200
	50	80	4000	2	1.01	15	180382	3.00	7200	7200	0.34	5.38	7200	7200
	50	80	4000	3	1.17	8	183081	3.71	7200	7200	0.45	5.98	7200	7200
	100	10	2000	1	0.00	0	2990986	0.96	7200	67	-	-	7200	-
	100	10	2000	2	0.00	0	2929535	1.08	7200	374	-	-	7200	-
	100	10	2000	3	0.00	0	2994928	0.83	7200	149	-	-	7200	-
	100	10	4000	1	0.02	0	2986108	0.75	7200	7200	-	-	7200	-
	100	10	4000	2	0.01	1	2928109	1.00	7200	7200	-	-	7200	-
	100	10	4000	3	0.10	0	2991843	0.67	7200	6017	-	-	7200	-
	100	80	2000	1	0.00	14	380060	4.51	7200	30	2.87	7.92	7200	3279
	100	80	2000	2	0.36	16	359222	2.90	7200	7194	4.17	7.35	7200	7200
	100	80	2000	3	0.58	12	369101	3.03	7200	7066	2.43	6.04	7200	7200
	100	80	4000	1	0.16	15	378273	4.08	7200	7200	-	-	7200	-
	100	80	4000	2	0.35	15	359262	2.74	7200	7200	2.71	6.47	7200	7200
	100	80	4000	3	0.43	13	369679	3.28	7200	7194	-	-	7200	-
					0.30	6		1.44	6505	5082	0.89	3.25	7200	6146

Table 6: B&C vs CplexFlow on $|V| = 50$ instances

					BS+H$_{Sup}$		B&C				CplexFlow			
f%	$\|K\|$	λ	Q	#	Δ%	t	obj	gap%	t	ttb	Δ%	gap%	t	ttb
90	50	10	1500	1	0.00	0	1737798	0.76	7200	34	-	-	7200	-
	50	10	1500	2	0.00	1	2130913	0.92	7200	75	-	-	7200	-
	50	10	1500	3	0.00	0	1994485	0.77	7200	71	-	-	7200	-
	50	10	3000	1	0.01	1	1732998	0.35	7200	7178	0.38	0.89	7200	0
	50	10	3000	2	0.01	1	2121931	0.41	7200	7200	-	-	7200	-
	50	10	3000	3	0.02	0	1988792	0.39	7200	7171	0.45	0.93	7200	1404
	50	80	1500	1	0.58	54	199780	2.25	7200	7200	0.26	3.75	7200	7200
	50	80	1500	2	0.75	44	209430	0.68	7200	7200	0.95	2.77	7200	7200
	50	80	1500	3	0.43	38	213850	1.23	7200	7162	1.15	3.26	7200	7200
	50	80	3000	1	0.84	54	199266	0.61	7200	7191	0.21	3.07	7200	7047
	50	80	3000	2	0.70	44	209059	0.20	7200	6817	0.10	1.35	7200	7063
	50	80	3000	3	0.55	38	213585	0.45	7200	6964	0.00	1.57	7200	6971
	100	10	1500	1	0.00	0	3919860	0.81	7200	1654	-	-	7200	-
	100	10	1500	2	0.00	0	3801784	0.75	7200	1486	-	-	7200	-
	100	10	1500	3	0.00	0	4385054	0.88	7200	2819	-	-	7200	-
	100	10	3000	1	0.00	0	3907498	0.50	7200	263	-	-	7200	-
	100	10	3000	2	0.00	0	3790026	0.45	7200	421	-	-	7200	-
	100	10	3000	3	0.00	0	4371270	0.57	7200	581	-	-	7200	-
	100	80	1500	1	0.02	60	397644	2.35	7200	7200	1.70	4.15	7200	3078
	100	80	1500	2	0.00	51	399830	2.82	7200	37	2.78	5.54	7200	4096
	100	80	1500	3	0.03	80	440429	2.67	7200	7014	1.14	3.80	7200	4324
	100	80	3000	1	0.22	57	396904	1.87	7200	7200	1.97	4.26	7200	1733
	100	80	3000	2	0.38	52	398264	2.26	7200	7200	1.19	3.80	7200	7200
	100	80	3000	3	0.08	76	439435	2.33	7200	7200	0.98	3.45	7200	2506
80	50	10	2000	1	0.01	1	1743001	0.99	7200	7176	-	-	7200	-
	50	10	2000	2	0.00	1	2130593	0.86	7200	7198	-	-	7200	-
	50	10	2000	3	0.01	1	1999149	0.97	7200	6909	-	-	7200	-
	50	10	4000	1	0.03	1	1742495	0.85	7200	7200	1.44	2.49	7200	2420
	50	10	4000	2	0.08	1	2128653	0.69	7200	7056	-	-	7200	-
	50	10	4000	3	0.04	1	1998461	0.88	7200	7200	-	-	7200	-
	50	80	2000	1	0.86	47	204770	4.01	7200	7200	5.71	10.84	7200	7200
	50	80	2000	2	0.07	54	214151	0.68	7200	7200	0.13	2.73	7200	7200
	50	80	2000	3	0.87	53	216408	0.44	7200	7200	0.29	3.28	7200	7118
	50	80	4000	1	0.79	47	204897	3.59	7200	7198	8.38	13.24	7200	3731
	50	80	4000	2	0.08	59	214118	1.03	7200	6858	0.79	4.48	7200	7178
	50	80	4000	3	0.86	57	216430	0.69	7200	7200	0.14	2.62	7200	7007
	100	10	2000	1	0.00	0	3927672	1.02	7200	1118	-	-	7200	-
	100	10	2000	2	0.00	0	3808624	0.93	7200	1168	-	-	7200	-
	100	10	2000	3	0.00	1	4389237	0.97	7200	1570	-	-	7200	-
	100	10	4000	1	0.00	0	3927456	1.01	7200	375	-	-	7200	-
	100	10	4000	2	0.00	0	3806717	0.88	7200	204	-	-	7200	-
	100	10	4000	3	0.00	1	4384496	0.86	7200	247	-	-	7200	-
	100	80	2000	1	0.00	72	406293	4.35	7200	42	3.04	7.44	7200	2665
	100	80	2000	2	0.59	120	415129	6.52	7200	7200	0.70	7.24	7200	6796
	100	80	2000	3	0.67	120	462180	7.25	7200	7200	4.75	11.49	7200	6604
	100	80	4000	1	2.56	120	402072	3.17	7200	7200	1.77	5.33	7200	7200
	100	80	4000	2	3.08	120	404480	3.41	7200	7200	1.30	5.11	7200	4583
	100	80	4000	3	2.27	120	450108	4.55	7200	7200	2.22	6.87	7200	5803
					0.36	34		1.62	7200	4868	1.63	4.66	7200	5279