The capacitated vehicle routing problem with reloads

Hervé Kerivin, Mathieu Lacroix, Ali Ridha Mahjoub, Alain Quilliot

LIMOS, CNRS UMR 6158, Université Blaise Pascal - Clermont-Ferrand II (France)
{herve.kerivin,ridha.mahjoub}@math.univ-bpclermont.fr, {mathieu.lacroix,alain.quilliot}@isima.fr

ABSTRACT

In this paper, we consider a variant of the transportation problem where any demand may be dropped off elsewhere than at its destination, picked up later by the same or another vehicle, and so on until it has reached its destination. We present two mixed integer linear programming formulations based on a space-time graph. We also develop a branch-and-cut algorithm for the problem and present some computational results.
Keywords: Transportation problem, mixed integer linear program, multicommodity flow, metric inequalities, branch-and-cut.

1. INTRODUCTION

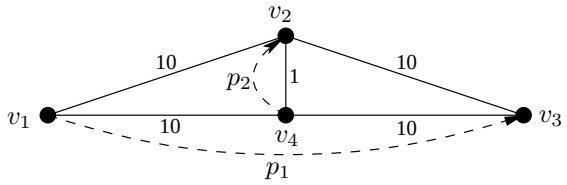
Transportation problems mainly consist of carrying through a given network some products, goods or people from their origins to their destinations. Because of important economic and ecologic stakes, operations research practitioners have been giving more and more attention to these problems. Many variants have then been introduced in order to match as close as possible the real-world applications. One of these variants allows demands to be unloaded and then reloaded in order to get better trips for the vehicles and to reduce the total cost. In this paper, we consider such a variant of the transportation problem.

This problem arises in a wide variety of freight and passenger transportation systems. This is the case for instance in postal services as described by Grünert and Sebastian in [8]. They present a problem encountered by the Deutch Post AG in its transportation chain, and that is as follows. Mail is first collected from customers and mail boxes and is then brought to the nearest Letter Mail Center (LMC). After being sorted, mail is sent from its origin LMCs to its destination ones. This transportation is performed by car, truck, aircraft or railway. Finally, after another sorting stage, mail is delivered to the given addresses by postmen. In this transportation problem, several optimization problems may be considered. The first step is thus a capacitated vehicle routing problem (introduced by Dantzig and Ramser in [4]), the second one for the ground problem is nothing but our problem and the last step is an arc routing problem (e.g., Chinese postman problem [6]).

We now precisely describe the problem studied in this paper. We consider a network specified by a set of nodes (e.g., cities) that are connected with each other by links (e.g., roads). Suppose that a set of demands is also given, each demand specified by an origin node, a destination node and a volume. The demands need to be carried through the network from their origins to their destinations using vehicles of a given fleet. All the vehicles have the same transportation capacity, and they all can begin and end their trips at any node of the network.
A demand can be unloaded at an intermediate node (i.e., a node different from its destination one), and can then be picked up later by the same or another vehicle. This unloading/picking-up process, called a *reload*, can be performed at any node of the network and can be repeated several times for a demand until its destination node is reached. Moreover, any demand can be splitted onto different trips and can be carried by several vehicles.
With each link of the network is associated a cost (e.g., fee) that corresponds to what must be paid to use the link. Yet, reload costs and time are neglected. The *Capacitated Vehicle Routing Problem with Reloads* (CVRPR) then consists of finding the vehicle trips so that all the demands are carried to their destinations, a vehicle is never overloaded, and the total cost is minimum.

The CVRPR can be seen as a relaxation of the standard Pickup and Delivery Problem (PDP). In fact in the latter, reloads are not allowed. (See [17] for a thorough description of the PDP.) As illustrated by the two following examples, this relaxation may lead to important savings. The first example shows that allowing reloads tends to bypass the limited vehicle capacity whereas in the second one, reloads are nothing but transshipments.

Fig. 1: Reloads with one vehicle



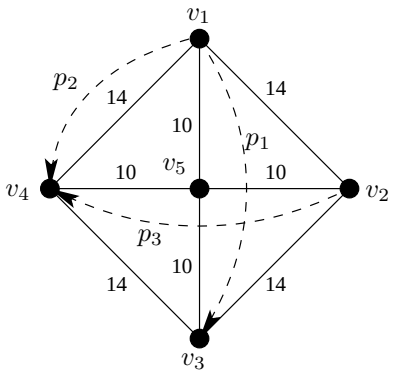
We first consider the network given in Figure 1 where each edge represents two opposite arcs. Suppose we have one vehicle and two demands p_1 and p_2 (represented by dashed arcs) such that the volume of each demand is lower than the capacity of the vehicle but that of both demands is greater than it.
An optimal solution of the PDP consists of making the vehicle start from v_1, bring p_1 to v_3 passing by v_4, then go back to v_4 where it loads p_2 and finally carry it to v_2. The associated cost is then 31. When reloads are allowed, an optimal solution can be as follows. The vehicle begins its trip by transporting p_1 from v_1 to v_4 where it unloads p_1 and loads p_2. It then carries p_2 from v_4 to v_2 and goes back directly to v_4. It finishes its trip by reloading p_1 at v_4 and conveying it to v_3. The associated cost is then 22.

In the second example shown in Figure 2, suppose we have

1-4244-0451-7/06/$20.00

two vehicles with an infinite capacity and three demands (represented again by dashed arcs). In an optimal solution without reloads, one vehicle conveys the demands p_1 and p_2 from v_1 to v_4 where p_2 is dropped off. Its finishes its trip by delivering p_1 at v_3. The second vehicle carries p_3 from v_2 to v_4 passing by v_5. The associated cost is 48. For the CVRPR, we obtain a cheaper solution. The first vehicle picks up p_1 and p_2 in v_1, goes to v_5 where it unloads p_2 and ends its trip at v_3 where it delivers p_1. The second vehicle uses the path v_2, v_5, v_4 as for the reloadless version problem but this time, it loads p_2 at v_5 and finishes its transportation from v_5 to v_4. The total cost is 40 and in this case, the reload is in fact a transshipment.

Fig. 2: Transshipment for one demand



Oertel [14] studied a variant of the PDP where reloads are permitted. But only unsplitable routing is allowed for the demands and reloads are only feasible on certain nodes, called hubs. He studied some properties of reloads and developped a local search algorithm for reload problems, taking into account additional constraints such as time windows.

Grünert and Sebastian [8] gave an integer linear programming formulation for the vehicle and request flow network design problem. This problem is close to the CVRPR but several additional constraints are taken into account such as time windows constraints and a limited capacity of reload at each node of the network (due to the time necessary to sort mail during the reloads). Moreover, reload costs and time are not neglected and it is imposed that each vehicle has to be unloaded whenever it reaches a location. To formulate this problem, Grünert and Sebastian used a multicommodity flow formulation on a space-time graph. They didn't purpose any algorithm to solve the problem.

The CVRPR can also be considered as a constrained version of the Minimum Cost Capacity Installation for Multicommodity network flows (MCCI), studied by Bienstock et al. in [3]. In fact, transportation capacity of vehicles can always be brought back to one (by dividing any demand volume by this capacity) since all the vehicles have the same capacity. The CVRPR is then a version of the MCCI in which the chosen capacities on arcs have to respect additional constraints in order to form feasible trips for the vehicles.

Despite the CVRPR is a relaxation of the PDP, it remains an NP-hard problem when there are several demands. It is easy to see it since the CVRPR is equivalent to the PDP if there is only one vehicle with an infinite capacity (in this case, reloads cannot be transshipments and they can then be removed by supposing that the demands stay in the vehicle instead). As the PDP is also NP-hard (the TSP with precedence constraints can be reduced to it), so is the CVRPR. On the other hand, if there is only one demand, the CVRPR can then be solved in polynomial time since it can be reduced to a sequence of shortest paths.

The remainder of the paper is organized as follows. In the next section, we introduce some notations and give, after having defined the construction of an auxiliary graph, two mixed integer linear programming formulations for the problem. We then present inequalities we add in order to strengthen the associated linear relaxations and we study their separation problems. We finally describe the branch-and-cut algorithm used to solve these two models and we present some computational results.

2. MIXED INTEGER LINEAR PROGRAMMING FORMULATIONS

Some notations and a more formal definition of the CVRPR are required at this point. To represent the network, we consider a directed graph $G = (V, A)$, called initial graph, where V corresponds to the set of vertices (i.e., nodes in the network) and A to the set of arcs (i.e., links in the network). For each arc $a \in A$, let $c_a \in \mathbb{R}_+$ be the cost for a vehicle to use the associated link, and let $l_a \in \mathbb{Z}_+$ be the time a vehicle need to go through a.
We denote by F the fleet of vehicles and by $B \in \mathbb{Z}_+$ their transportation capacity. Let P be the set of demands that have to be carried through the network. With every demand $p \in P$, we associate a triplet (o^p, d^p, q^p) where $o^p \in V$ represents its origin node, $d^p \in V$ its destination node and $q^p \in \mathbb{Q}_+$ its volume.

Any solution of the CVRPR has to satisfy what we call *precedence conditions*. In fact, because of the reload policy, part of a demand $p \in P$ may be dropped off at a node $v \in V$ that is different from its destination d^p and then picked up later. Therefore, the vehicle that carries this part of p on the leg started at v has to pass by v once the part of the demand is arrived at v, that is the leg ended at v is completed. To handle such conditions, we consider an auxiliary directed graph that is based on a space-time graph. (Similar graphs can be found in [1] where it is referred as time-expanded network and in [8].) We then need to be given a completion time limit $T \in \mathbb{Z}_+$ that corresponds to the latest any demand of P arrives at its destination node. (Remark that T can be as big as necessary to keep the initial solution space.)

2.1. Space-time graph

This subsection is devoted to the construction of the space-time graph, denoted by $G_{st} = (V_{st}, A_{st})$, from the initial graph G. For each vertex $v \in V$, we associate $T + 1$ vertices $v_0, v_1, \ldots, v_T$ in G_{st}. The vertex $v_t \in V_{st}$ represents $v \in V$ at time $t \in \{0, \ldots, T\}$. (Note that time is considered discrete.) We then consider a first arc set of G_{st}, $A_T = \{(v_t, v_{t+1}) \mid v \in V,\ t \in \{0, \ldots, T-1\}\}$, where an arc $a = (v_t, v_{t+1}) \in A_T$ corresponds to a vehicle or a demand that stays at a node for one time unit. We also con-

sider a second arc set $\tilde{A} = \{(u_t v_{t+l_{(u,v)}}) \mid (u,v) \in A,\ t \in \{0,\ldots,T-l_{(u,v)}\}\}$, where an arc $a = (u_t, v_{t'}) \in \tilde{A}$ corresponds to a vehicle that goes from node u at time t to node v at time t' with $t' = t + l_{(u,v)}$. The arcs in A_T have a zero cost whereas an arc $a = (u_t, v_{t'}) \in \tilde{A}$ has its cost c_a equal to $c_{(u,v)}$.
Graph G_{st} is then defined by the vertex set $V_{st} = \{v_t \mid v \in V, t \in \{0,\ldots,T\}\}$ and the arc set $A_{st} = A_T \cup \tilde{A}$. This graph has $|V|(T+1)$ vertices and up to $T(|A|+|V|)$ arcs.

2.2. Auxiliary graph

We now define an auxiliary graph $G' = (V', A')$ obtained from G_{st} by adding two additional vertices O and D and $2|V|$ arcs as follows. Vertices O and D respectively represent the origin and the destination of the vehicle trips. (Remark that since all the vehicles begin and end their trips at any node of the network, O and D can then be seen as dummy depots.) Without loss of generality, we force all the vehicles of the fleet to start their trip at time 0 and finish it at time T. We then add arcs (O, v_0) and (v_T, D) for all vertices $v \in V$, both having a zero cost. Let $A^O = \{(O, v_0) \mid v \in V\}$ and $A^D = \{(v_T, D) \mid v \in V\}$. We have $V' = V_{st} \cup \{O, D\}$ and $A' = A_{st} \cup A^O \cup A^D$.

Below is an example of the construction of the auxiliary graph G' from the initial graph $G = (V, A)$ given in Figure 3. We suppose that all the arcs $a \in A$ have a duration time l_a equal to one and T equals two. Starting with an initial graph having 3 vertices and 3 arcs, we end up with an auxiliary graph that has 11 vertices and 18 arcs as shown in Figure 4.

Fig. 3: Initial graph G

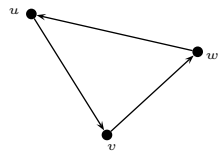
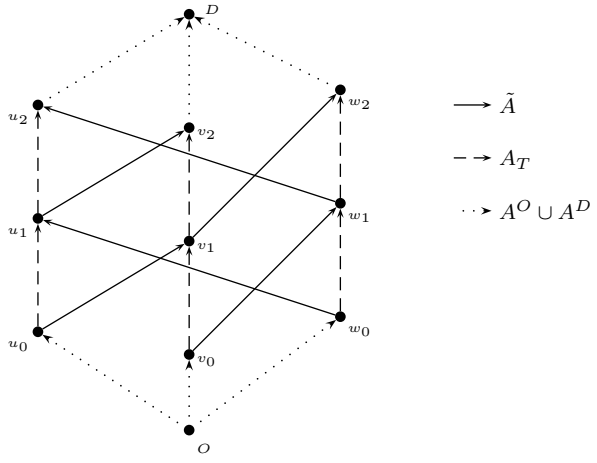


Fig. 4: Auxiliary associated graph G'



2.3. Multicommodity flow based formulation

In this formulation, we consider a multicommodity flow to represent the routing of the demands in the network and a flow to model the trips of the vehicles. We then have two types of variables that are:

- $y \in \mathbb{Z}_+^{|A'|}$ where y_a represents the number of vehicles passing through arc $a \in A'$,
- $x \in \mathbb{R}_+^{|A_{st}|\times|P|}$ where x_a^p represents the amount of demand $p \in P$ carried on the arc $a \in A_{st}$.

For a vertex $v \in V_{st}$ and a demand $p \in P$, the number b_v^p defined by

$$b_v^p = \begin{cases} q^p & \text{if} \quad v = o_0^p, \\ 0 & \text{if} \quad v \neq o_0^p, d_T^p, \\ -q^p & \text{if} \quad v = d_T^p, \end{cases}$$

represents the supply/demand associated to vertex v with respect to demand p. We remark that for a given $p \in P$, there are exactly two vertices of V_{st} for which b_v^p is non null. These two vertices are o_0^p and d_T^p, that is, the origin node of p at time 0 and the destination one at time T respectively. The correctness of the definition of b_v^p is implied by the equivalence between demand p (or part of it) staying for some time at a node of the network and a path using only arcs of A_T in G'.

The CVRPR can then be formulated as a mixed integer linear program using an arc-node based approach [1] as follows.

$$\min \sum_{a \in A_{st}} c_a y_a$$

s.t.

$$\sum_{a \in A^O} y_a \leq |F| \tag{1}$$

$$\sum_{a \in \delta^+(v)} y_a - \sum_{a \in \delta^-(v)} y_a = 0 \quad \forall\, v \in V_{st}, \tag{2}$$

$$\sum_{\substack{a \in \delta^+(v) \\ a \in A_{st}}} x_a^p - \sum_{\substack{a \in \delta^-(v) \\ a \in A_{st}}} x_a^p = b_v^p \quad \forall\, p \in P,\ \forall\, v \in V_{st}, \tag{3}$$

$$\sum_{p \in P} x_a^p \leq B y_a \qquad \forall\, a \in A_{st} \setminus A_T, \tag{4}$$

$$x_a^p \geq 0 \qquad \forall\, a \in A_{st},\ \forall\, p \in P, \tag{5}$$

$$y_a \geq 0 \qquad \forall\, a \in A', \tag{6}$$

$$y_a \text{ integer} \qquad \forall\, a \in A', \tag{7}$$

where $\delta^+(v)$ (resp. $\delta^-(v)$) denotes the set of arcs of A' having $v \in V'$ as a tail (resp. head). The objective fuction states that the total vehicle-related cost must be minimized. (We remark that this objective function can easily be extended if costs proportionate to the demand amounts carried on arcs are considered.) Constraints (1) force the number of used vehicles to be at most $|F|$. Constraints (2) (resp. (3)) are the flow conservation constraints associated

with the vehicles (resp. demands). Constraints (4) impose the amount of the demands carried on an arc of $A_{st}\backslash A_T$ to be no more than the total capacity of the vehicles passing through this arc.

2.4. Metric constraints based formulation

In the formulation given in the previous subsection, we determine the routing of the demands even though the value of the objective function only depends on the vehicle trips and no additional constraints on the demand routing are considered. Therefore, it would be enough to only focus on determining the vehicle trips that allow a feasible routing of the demands. This can be achieved by considering no other variables than y_a for all $a \in A'$ and replacing constraints (3) and (4) by the so-called metric constraints as described as below.

Metric constraints have been introduced independently by Iri [10] and Onaga and Kakhuso [15]. They permit to check if there exists a feasible multicommodity flow for a given graph when demands and capacities are fixed. Their result, known as Japonese theorem, can be briefly presented as follows. Let $\bar{G} = (\bar{V}, \bar{A})$ be a complete directed graph and $w \in \mathbb{R}_+^{|\bar{A}|}$ (resp. $r \in \mathbb{R}_+^{|\bar{A}|}$) be the capacity (resp. demand) vector indexed on the arcs of $\bar{A}$. The capacity vector w allows the transportation of the demands of r if and only if all the metric constraints

$$(w - r)^T \pi \geq 0 \quad \forall\, \pi \in Met_n \tag{8}$$

are satisfyied where
$Met_n = \{\pi \in \mathbb{R}_+^{\bar{A}} \mid \pi_{ik} + \pi_{kj} - \pi_{ij} \geq 0 \,\forall i \neq j \neq k \neq i\}$ is the *metric cone*.

For our problem, the metric cone is the one induced by the complete graph on V_{st}, capacity and demand vectors are given by

$$w_a = \begin{cases} +\infty & \text{if } a \in A_T, \\ By_a & \text{if} a \in A_{st}\backslash A_T, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$r_a = \begin{cases} q^p & \text{if } a = (o_0^p, d_T^p) \text{ for some } p \in P, \\ 0 & \text{otherwise,} \end{cases}$$

for all $a \in V_{st} \times V_{st}$. Using metric constraints (8) instead of constraints (3) and (4), we now introduce the following integer linear formulation for the CVRPR:

$$\min\{ \sum_{a \in A_{st}} c_a y_a \mid y \text{ satisfies } (1), (2), (8), (6), (7)\} \tag{9}$$

This linear program contains less variables than the first one (i.e., $|A'|$ versus $|A'| + |A_{st}| \times |P|$), but it has an exponential number of constraints whereas the first model contains a polynomial number of constraints. As it will be shown in Section 3, the exponential number of metric constraints can be tackled in polynomial time.

Furthermore, once we have an optimal solution of (9), determining the routing of the demands (i.e., the variables x_a^p for all $a \in V_{st}$ and for all $p \in P$ in the first formulation) can be performed in polynomial time. It is actually nothing but a continuous multicommodity flow problem which is a well known polynomially-solvable problem [1].

2.5. Formulation strengthening

In this section, we introduce constraints known as bipartition constraints that we will use in our branch-and-cut algorithm to strengthen both linear relaxations of the CVRPR. These constraints have been already considered in various optimization problems. (See [2, 3, 5, 13] for instance.)

These new constraints are based on the notion of cut in G_{st}. Given a vertex subset $W \subseteq V_{st}$ so that $W \neq \emptyset$, $W \neq V_{st}$, the cut induced by W is denoted by $\delta^+(W)$ and corresponds to the set of arcs of A_{st} having their tails in W and their heads in $V_{st}\backslash W$. To clearly present the bipartition constraints, we introduce the following notations. Let $z \in \mathbb{R}^{A'}$ be a vector and $X \subseteq A'$ a subset of arcs, we will write $z(X)$ for $\sum_{x \in X} z(x)$. For a vertex subset $W \subseteq V_{st}$, let us denote by $q[W, V_{st}\backslash W]$ the total volume of the demands of P having their origin vertex in W and their destination one in $V_{st}\backslash W$.

Theorem 1 *Let $W \subseteq V_{st}$ induce a cut $\delta^+(W)$ so that $\delta^+(W) \cap A_T = \emptyset$. The bipartition constraint*

$$y(\delta^+(W)) \geq \left\lceil \frac{q[W, V_{st}\backslash W]}{B} \right\rceil \tag{10}$$

is valid for the CVRPR.

Proof: The constraint $By(\delta^+(W)) \geq q[W, V_{st}\backslash W]$ is obviously valid for the CVRPR. In fact, it expresses that the whole vehicle capacity of the arcs of $\delta^+(W)$ must exceed the whole demand from W to $V_{st}\backslash W$. Since y is integer, dividing the two members of this inequality by B and rouding-up the right-hand side yields (10). □

We remark that we only consider cuts that do not intersect A_T because capacity constraints do not apply for these arcs. (In the first formulation, there are no constraints (4) for these arcs and in the second one, we consider an infinite capacity on them.) This comes directly from the fact that any demand (or part of it) can stay without any conditions at any node of the network.

The number of bipartition constraints is exponential. In the next section, we show how to manage these constraints in our branch-and-cut algorithm.

3. SEPARATION PROBLEM

One of the most important parts of an efficient branch-and-cut algorithm is the so-called separation problem than can be described as follows.
Given a constraint system $Ax \leq b$ based on $\mathbb{R}^n$ and a point x of $\mathbb{R}^n$, the *separation problem* associated with this system consists of deciding whether all the constraints of the system are satisfyied by x and if not, of finding a constraint violated by x. Grötschel, Lovasz and Schriver [7] have shown that if the separation problem of a constraint system $Ax \leq b$ is polynomial, then an optimization problem over this system is polynomial even if the number of constraints of the system $Ax \leq b$ is exponential.

In this section, we consider the separation problems for the metric constraints (8) and the bipartition constraints (10) with respect to a given vector $\bar{y} \in \mathbb{R}^{A'}$.

3.1. Separation of metric constraints

The separation problem for metric constraints can be solved in polynomial time. In fact, it can be reduced to the following continuous linear program:

$$\begin{aligned} &\min(w-r)^T\pi \\ &\text{s.t.} \\ &\pi \in Met_n, \end{aligned}$$

where Met_n, w and r are defined as in Subsection 2.4. Let π^* be an optimal solution of this linear program. If the objective value associated with π^* is negative, the metric constraint $(w-r)^T\pi^* \geq 0$ is then violated by $\bar{y}$. Otherwise, we can assert that $\bar{y}$ satisfies all the metric constraints.

3.2. Separation of bipartition constraints

The separation problem associated with bipartition constraints (10) is NP-hard in general. We can notice that this problem is also NP-hard without considering the rounding-up of the right-hand side of (10). In fact, Barahona [2] showed that the max-cut problem can be reduced to this problem.
In our branch-and-cut algorithm, we then need to use heuristics to separate the bipartition constraints. We have devised the three following separation heuristics that are based on previous works [3, 5]. In the description of those heuristics, W will always represent a vertex subset of V_{st} so that $\delta^+(W) \cap A_T = \emptyset$.

The first one is nothing but the so-called n-Cut heuristic that was developed by Bienstock et al. for the MCCI [3]. This heuristic works as follows. For any demand $p \in P$, we check whether there exists a path from o_0^p to d_T^p in the graph G' where we only consider the arc set $A_T \cup \{a \in \tilde{A} \mid \bar{y_a} > 0\}$. This can obviously be performed using any kind of search algorithm. If there is not such a path, it is straightforward to see that there exists a violated constraint (10). This constraint is indeed induced by a vertex set W so that $o_0^p \in W$, $d_T^p \in V_{st}\backslash W$ and the vertices of V_{st} reachable from o_0^p belong to W. ($\delta^+(W) \cap A_T = \emptyset$ is guaranteed because all the arcs of A_T are considered.)
If a path from o_0^p to d_T^p is found for all the demands $p \in P$, we then randomly pick out some vertices of V_{st} to form a set W. We then check if the bipartition constraint associated with W is violated or not.

The second heuristic is based on the so-called n-partition heuristic devised by Bienstock et al. in [3]. We only consider the case n=2 since we seek violated bipartition constraints (10). We start with a randomly chosen demand $p \in P$ and the two vertex subsets of V_{st}, $V_1 = \{o_t^p \mid t = 0, \ldots, T\}$ and $V_2 = \{d_t^p \mid t = 0, \ldots, T\}$. We then iteratively assign the vertices of $V_{st}\backslash(V_1 \cup V_2)$ to either V_1 or V_2 as described below. (At the end of the heuristic, we will consider the bipartition constraint induced by $W = V_1$.) At each iteration, we randomly select a vertex $v \in V\backslash\{o^p, d^p\}$ that has not been considered yet. We then assign up to $T+1$ vertices of $\{v_t \mid t = 0, \ldots, T\}$ to V_1 (the other vertices of $\{v_t \mid t = 0, \ldots, T\}$ being assigned to V_2) in order to obtain sets V_1 and V_2 so that $\delta^+(V_1) \cap A_T = \emptyset$ and $B\bar{y}(\delta^+(V_1) \cap \delta^-(V_2)) - q[V_1, V_2]$ is minimum. ($\delta^-(V_2)$ denotes the set of arcs of A_{st} entering V_2.) Once all the vertices of $V\backslash\{o^p, d^p\}$ have been considered, we check wheither the bipartition constraint induced by $W = V_1$ is violated.

The last heuristic is an extension of the one developed by Gabrel et al. in [5]. Given a demand $p \in P$, let W be a randomly chosen set so that $o_0^p \in W$ and $d_T^p \in V_{st}\backslash W$. This heuristic consists of iteratively switching vertices (except o_0^p and d_t^p for all $t = 0, \ldots, T$) in-between W and $V_{st}\backslash W$ in order to increase the value of $p(W) = \lceil q[W, V_{st}\backslash W]/B \rceil / \bar{y}(\delta^+(W))$. This process stops when no switchings increase the ratio $p(W)$.
To find the switching, we compute for each vertex $v \in V\backslash\{o^p, d^p\}$ that has not been considered yet the value $\alpha(v)$. The latter corresponds to the assignment of the vertices v_t for all $t = 0, \ldots, T$ to the subsets W and $V_{st}\backslash W$ that maximizes $p(W)$. (The assignment of the other vertices does not change.) We then only modify the assignment of the vertices $\{v_t \mid t = 0, \ldots, T\}$ associated with the vertex v for which $\alpha(v)$ is maximum. We repeat it until all the vertices of $V\backslash\{o^p, d^p\}$ have been considered. At this point, we obtain the new subset W.
Once no switching increases the ratio, we test if the bipartition constraint induced by the obtained set W is violated. We repeat this algorithm for all the demands $p \in P$.

4. BRANCH-AND-CUT ALGORITHM

In this section we describe a branch-and-cut algorithm for the CVRPR based on the formulations given in Section 2. We assume the reader to be familiar with this method. If not, one can refer to [16] for a description of this technique. We will also present some computational results.

4.1. Solving method

To start the optimization for the first model, we consider its linear relaxation. For the metric constraint based model, the optimization starts with inequalities (1) and (2). If an optimal solution of the linear relaxation of the problem is not optimal, then the branch-and-cut algorithm tries to generate violated bipartition constraints using the heuristics described above. For the second model, if no constraint of this type is found, the algorithm then generates violated metric inequalities, if there is any. The branching procedure is based on the *strong branching* operation introduced in CPLEX 7.5 [9].

4.2. Computational results

The branch-and-cut algorithm is implemented in C++ using the free software COIN-OR [12]. We use BCP module to manage the branching tree, CLP module as the linear solver and OSI module as the interface between CLP and BCP. It is tested on a Pentium IV 3.2 Ghz with 1GB of RAM and a running time limited to 5 hours.
The instances consist in randomly generated complete graphs with arc costs equal to the rounded euclidian distances and arc duration time equal to one. The tests were performed with $T = 5$.

Table 1 reports the results obtained using our algorithm. For each test, we solve three instances with different demands. Each of the instances is solved with four random number generators. Each line of Table 1 gives the average results for one test.

In Table 1 are specified the number of vertices $|V|$, the number of demands $|P|$, the number of vehicles $|F|$, the number of generated bipartition (resp. metric) constraints bip (resp. met). It is also indicated the gap Gap between the best upper bound and the lower bound obtained at the root node of the branch-and-cut tree, the number Opt of instances solved to optimality over the number of tested instances, the number No of nodes of the branch-and-cut tree and the CPU time in seconds.

Tab. 1: Results for the CVRPR

|V|	|P|	|F|	Bip	Met	Gap	Opt	No	CPU
5	4	1	19.00	-	0.00	12/12	5.67	0.17
6	8	4	228.67	-	0.61	12/12	873.17	225.46
7	9	5	272.50	-	0.00	12/12	850.00	421.30
8	6	2	85.92	-	1.19	12/12	10.33	3.04
9	11	6	92.33	-	0.37	6/12	4.00	7.40
10	8	3	158.33	-	2.01	12/12	19.67	15.78
5	4	1	17.92	0.00	26.71	12/12	22.33	0.51
6	8	4	87.42	4.78	0.61	12/12	408.50	31.85
7	9	5	116.50	12.40	0.51	12/12	449.67	118.90
8	6	2	68.92	3.36	32.44	12/12	237.00	339.94
9	11	6	457.58	1673.00	2.08	12/12	2161.33	5112.18
10	8	3	144.91	1.91	30.56	11/12	544.09	6826.85

The first part of Table 1 summarizes the results for the arc-node model. The second one concerns the metric constraint based model. Each test is performed for both models. We remark that for most of the tests, the whole set of instances is solved to optimality in a reasonnable computational time. We also remark that for both models the algorithm generates an important number of bipartition inequalities. Although the instances are relatively small, some of them could not be solved to optimality within the time limit. For instance, for the arc-node model and graphs on 9 nodes, only half of the instances have been solved to optimality. However, with the second model, all the instances have been solved. Also note that the instances on 10 nodes with the first model have been solved in a few seconds whereas for the same instances, in the second model, an average of two hours was needed to solve them. Moreover, one of them could not even be solved within the time limit. So one cannot decide which of the two models would be more efficient. But, as these experimental results are still in a very preliminary version, more investigations may help to compare the two models. This is our research direction in the near future [11].

5. CONCLUSION

In this paper, we have proposed two models for the capacitated vehicle routing problem with reloads. We have developed a branch-and-cut algorithm for solving both models. These can be extended in order to take into account more constraints such as a vehicle depot or vehicles with different capacities.
Moreover, it would be interesting to develop a model which does not use time indexation. Such a model would be useful for solving larger instances. This is also our aim for future work.

REFERENCES

[1] Ahuja R., T. Magnanti and J. Orlin, "Networks flows : theory, algorithms and applications", *Englewood Cliffs, NJ : Prentice Hall*, 1993.

[2] Barahona F., "Network design using cut inequalities", *SIAM Journal on Optimization*, Vol. 6, pp823-837, 1996.

[3] Bienstock D., S. Chopra, O. Günlünk and C. Tsai, "Minimum cost capacity installation for multicommodity network flows", *Mathematical Programming*, Vol. 81, pp177-199, 1995.

[4] Dantzig G.B. and R.H. Ramser, "The truck Dispatching Problem", *Management Science*, Vol. 6, pp80-91, 1959.

[5] Gabrel V., A. Knippel, M. Minoux, "Exact solution of multicommodity network optimization problems with general step cost functions", *Operation Research Letter*, Vol. 25, pp15-23, 1999.

[6] Gendrau M. and G. Laporte, "Arc routing problems, part I : The Chinese Postman Problem", *Operations Research*, Vol. 43, pp231-242, 1995.

[7] Grötschel M., L. Lovasz and A. Schrijver, "Geometric algorithms and combinatorial optimization", *Springer-Verlag*, 1985.

[8] Grünert T. and H.-J. Sebastian, "Planning models for long-haul operations of postal and express shipment companies", *European Journal of Operational Research*, Vol. 122, pp289-309, 2000.

[9] ILOG CPLEX. "Reference manual", *http://www.ilog.com/products/cplex*, 2003.

[10] Iri M., "On an extension of the maximum-flow minimum-cut theorem to multicommodity flows", *J. Operations Research Soc. of Japan*, Vol. 13, No. 3, 1971.

[11] Kerivin H., M. Lacroix, A. R. Mahjoub and A. Quilliot, "Polyhedron study of a transportation problem with reloads", *LIMOS, Technical Report* 2006, in preparation.

[12] Lougee-Heimer R., "The Common Optimization INterface for Operations Research", *IBM Journal of Research and Development*, Vol. 47, No. 1, pp57-66, 2003.

[13] Magnanti T., P. Mirchandani and R. Vachani, "Modeling and solving two facility capacitated network loading problem", *Operation Research*, Vol. 43, No. 1, pp142-157, 1995.

[14] Oertel P., "Routing with reloads", *PhD Thesis*, Köln, 2000.

[15] Onaga K. and O. Kakhuso "On feasibility conditions of multicommodity flows networks", *Transactions on circuit theory*, Vol. 4, pp425-429, 1971.

[16] Padberg M., G. Rinaldi, "A branch-and-cut algorithm for the resolution of large-scale symmetric travelling salesman problems", *SIAM Review*, Vol. 33, pp60-100, 1991.

[17] Savelsberg M. W. P. and M. Sol, "The general pickup and delivery problem", *Transportation Science*, Vol. 29, pp17-29, 1995.