Vehicle Routing Problem with Elementary Shortest Path based Column Generation

Alain Chabrier
ILOG
Gobelas, 21,
28023 Madrid
achabrier@ilog.fr

April 10, 2003

Abstract

The usual column generation model for a Vehicle Routing Problem involves an elementary shortest-path subproblem. The worst-case complexity of the known algorithms for this problem being too high, the elementary-path constraint is usually relaxed. Indeed, as each customer must be visited exactly once, the two problems with and without the elementary-path constraint have the same optimal integer solutions. In this article, we propose one theoretical and several practical improvements to the algorithm for elementary paths. We obtain better lower bounds and pruning of the search tree, and these improvements allowed us to find an exact solution to 17 instances of the Solomon benchmark suite which were previously open.

1 Introduction

Many academic publications have been dedicated already to Vehicle Routing Problems (VRPs). The problems from this family mimic the real-life logistic problem of minimizing the cost of distributing goods from a warehouse to a set of destinations by means of several vehicles. VRP data are a set of nodes to be visited, a set of vehicles initially located at some special node (called the *depot*), and that must get back to it at the end of the route, and a distance scheme, associating to each pair of nodes the distance to travel from one to the other. The objective is to find a set of routes (i.e., ordered sequences of visits) for the vehicles, covering all the visits and doing so with the minimal total length travelled. The simplest extension uses a fixed capacity to limit the quantity of goods deliverable by a specific vehicle. Other extensions include a minimum and maximum time for each node constraining the moment at which it can be visited. Several other extensions are commented in the literature: minimization of the number of vehicles before total distance, use of several depots. In this paper, we worked on the Vehicle Routing Problem with Time Windows (VRPTW).

Among the studies of VRPs, two categories can be identified. On the one hand, heuristic methods are used to find solutions of acceptable quality quickly, and are usually based on local search techniques ([RT95, CLM01, HG99, GTA99]) sometimes in hybrid cooperation with constraint programming ([DFS$^+$00, RGP99, Sha98, KPS00, BH01]). On the other hand, exact methods focus on finding an optimal solution. Column Generation techniques are part of those exact methods ([DDS92, CR99, Lar99, KLM01]). The original linear program is decomposed by means of the Dantzig-Wolfe decomposition into a linear restricted master problem and a pricing subproblem. The master problem then becomes a partitioning problem with binary variables, and the pricing subproblem is a constrained shortest-path problem in charge of generating new promising columns. In the original VRP, each customer has to be visited once. Hence, the routes should not include cycles, so that the subproblem of the decomposed version that should be solved is the Elementary Shortest-Path Problem with Resource Constraints and Time Windows (ESPRCTW). This cycle constraint is usually relaxed and instead, a Shortest Path with Resource Constraints and Time Windows (SPRCTW) is used. In that context, it can be shown that the optimal integer solution of this column generation model with the cycle constraint relaxed will contain only elementary routes. The main reason for relaxing this cycle constraint is that more powerful algorithms are available to solve the SPRCTW than to solve the ESPRCTW. In fact, pseudo-polynomial algorithms are available for the SPRCTW, whereas in contrast, it has been shown that ESPRCTW is NP-hard [Dro94]. To our knowledge, an ESPRCTW algorithm for VRP is proposed only in [FDGG02]. In [RGP02], the elementary sub-problem is solved using constraint programming.

In this article, we present several advantages of ESPRCTW, such as the better quality of the lower bounds obtained with the relaxed restricted master problem, and we present several techniques to reduce the impact in efficiency of taking the cycle constraint into account. We give some promising results obtained on a well known benchmark suite.

The article is organized as follows. In the next section, we formally introduce the Vehicle Routing Problem as well as its linear programming model, and we show the model resulting from the Dantzig-Wolfe decomposition. In Section 3, we present our method, introducing the column generation scheme and the elementary shortest-path subproblem. Then, in Section 4, we discuss the possible differences in quality between our bounds and those obtained by other methods. In Section 5, we present our ESPRCTW implementation along with several practical enhancements. Finally, we present computational results in Section 6.

2 The VRP decomposed model

In this section, we introduce two different models for the VRP. In the Vehicle Routing Problem with Time Windows, K vehicles with respective maximum capacities C_k are initially available at a specific position referred as the *depot*. N visits must be performed. At each visit i, a quantity q_i of material must be delivered. The whole quantity must be delivered at the same time that must be in a time window interval defined by $[a_i, b_i]$. A is a set of arcs between those $N+1$ sites, and a function d, identical for all the vehicles, defines the distance corresponding to each arc $(i,j) \in A$. The distance d is also used to measure time. The routes must start and end at two dummy visits, 0 and $N+1$, representing the depot. The objective is to find the minimal distance set of routes for the K vehicles to deliver the quantities to all the visits.

2.1 MIP model

The compact mixed integer model involves binary variables x_{ijk} indicating whether vehicle k uses the arc from i to j, and variables s_{ik} indicating the time at which vehicle k reach visit i. Constraints ensure the routes correspond to a valid path (i.e. total vehicle capacity respected and each visit covered exactly once).

The MIP model is thus:

$$min \sum_{k=1}^{K} \sum_{(i,j) \in A} d(i,j) x_{ijk} \tag{1}$$

$$\sum_{j \in \delta^+(0)} x_{0jk} = 1, \qquad \forall k \in \{1, ..., K\} \tag{2}$$

$$\sum_{i \in \delta^-(j)} x_{ijk} - \sum_{i \in \delta^+(j)} x_{jik} = 0, \qquad \forall k \in \{1, ..., K\}, \forall j \in \{1, ..., N\} \tag{3}$$

$$\sum_{i \in \delta^-(N+1)} x_{i,N+1,k} = 1, \qquad \forall k \in \{1, ..., K\} \tag{4}$$

$$\sum_{i \in S} \sum_{j \in S, j \neq i} x_{ijk} \leq |S| - 1 \qquad \forall k \in \{1, ..., K\}, \forall S \subset X, 1 < |S| < N \tag{5}$$

$$\sum_{k=1}^{K} \sum_{j \in \delta^+(i)} x_{ijk} = 1, \qquad \forall i \in \{1, ..., N\} \tag{6}$$

$$\sum_{i=1}^{N} q_i \sum_{j \in \delta^+(i)} x_{ijk} \leq C_k, \qquad \forall k \in \{1, ..., K\} \tag{7}$$

$$s_{ik} + d(i,j) - K(1 - x_{ijk}) \leq s_{jk}, \qquad \forall (i,j) \in A, \forall k \in \{1, ..., K\} \tag{8}$$

$$a_i \leq s_{ik} \leq b_i, \qquad \forall i \in \{1, ..., N\}, \forall k \in \{1, ..., K\} \tag{9}$$

$$x_{ijk} \in \{0, 1\}, \qquad \forall k \in \{1, ..., K\}, \forall (i,j) \in A \tag{10}$$

Constraints (2-4) define the flow for vehicle k, constraints (5) eliminate the possible subtours, constraints (6) ensure each visit is made by at most one vehicle, and constraints (7) ensure that capacities of vehicles are respected. Constraints (8) and (9), defines time windows. Finally, the objective (1) minimizes the sum of the distance of the arcs in use.

Note that the number of constraints on subtours is exponential so that they are added dynamically during the search, using a Branch-and-Cut approach.

2.2 Dantzig-Wolfe decomposition

The Dantzig-Wolfe decomposition ([DW60]) is then applied to this MIP model. We consider the matrix B_k corresponding to the constraints (2-5) and (7) for a given k. The constraint (6) is kept apart. The polytope defined by B_k admits as extreme points the set Ω_k of the valid paths for the vehicle k. We can thus write one of its points as a linear combination of those paths. To do so, we use coefficients x^p_{ijk}, the value of which is 1 if arc (i, j) belongs to the path corresponding to the extreme point λ_k^p, and 0 otherwise. For each $k \in \{1, ..., K\}$, a solution x_{ijk} can thus be rewritten as follows:

$$\begin{aligned} x_{ijk} = \sum_{p \in \Omega_k} x_{ijk}^p \lambda_k^p, \forall (i,j) \in A, \\ \sum_{p \in \Omega_k} \lambda_k^p = 1, \forall k \in \{1, ..., K\} \\ \lambda_k^p \geq 0, \forall p \in \Omega_k \end{aligned}$$

We can then define c_k^p as the cost of route p for vehicle k. Also, positive integers a_{ik}^p indicate the number of times visit i is made by vehicle k for the route p.

$$\begin{aligned} c_k^p &= \sum_{(i,j) \in A} c(i,j) x_{ijk}^p, \qquad \forall k \in \{1, ..., K\}, \forall p \in \Omega_k \\ a_{ik}^p &= \sum_{j \in \delta^+(i)} x_{ijk}^p, \qquad \forall i \in \{1, ..., \}N, \forall k \in \{1, ..., \}K, \forall p \in \Omega_k \end{aligned}$$

If we substitute these expressions into the original model, we obtain the decomposed model:

$$\begin{aligned} min \sum_{k=1}^{K} \sum_{p \in \Omega_k} c_k^p \lambda_k^p \\ \sum_{k=1}^{K} \sum_{p \in \Omega_k} a_{ik}^p \lambda_k^p = 1, &\qquad \forall i \in \{1, ..., N\} \\ \sum_{p \in \Omega_k} \lambda_k^p = 1, &\qquad \forall k \in \{1, ..., K\} \\ \lambda_k^p \geq 0, &\qquad \forall k \in \{1, ..., K\}, \forall p \in \Omega_k \end{aligned}$$

The capacity, subtour-elimination and time window constraints are integrally moved to the subproblems defining the validity of the extreme-point paths. That change can be formulated as follows:

$$
\begin{aligned}
min \sum_{(i,j)\in A} d(i,j)x_{ij} & \\
\sum_{j\in\delta^+(0)} x_{0j} = 1, & \\
\sum_{i\in\delta^-(j)} x_{ij} - \sum_{i\in\delta^+(j)} x_{ji} = 0, & \qquad \forall j \in \{1,...,N\} \\
\sum_{i\in\delta^-(N+1)} x_{i,N+1} = 1, & \\
\sum_{i\in S}\sum_{j\in S, j\neq i} x_{ij} \le |S|-1 & \qquad \forall S \subset X, 1 < |S| < N \\
\sum_{i=1}^{N} q_i \sum_{j\in\delta^+(i)} x_{ij} \le C_k, & \\
s_i + d(i,j) - K(1-x_{ij}) \le s_j, & \qquad \forall (i,j) \in A \\
a_i \le s_i \le b_i, & \qquad \forall i \in \{1,...,N\}, \\
x_{ij} \in \{0,1\}, & \qquad \forall (i,j) \in A
\end{aligned}
$$

We recognize here the problem that consists of finding elementary paths from depot to depot and respecting the capacity constraint.

2.3 Identical vehicles

When vehicles are identical (i.e., they have the same total capacity), some parts of the model can be compacted:

$$\lambda^p = \sum_{k=1}^{K} \lambda_k^p, \forall p \in \Omega = \bigcup_{k=1}^{K} \Omega_k$$

and we obtain:

$$
\begin{aligned}
min \sum_{p\in\Omega} c^p \lambda^p & \\
\sum_{p\in\Omega} a_i^p \lambda^p = 1, & \qquad \forall i = 1,...,N \qquad (11) \\
\sum_{p\in\Omega} \lambda^p = K, & \qquad (12) \\
\lambda^p \ge 0, & \qquad \forall p \in \Omega
\end{aligned}
$$

This decomposed model contains a huge number of possible path variables. If N is the number of visits, the number of theoretical path variables can grow up to $N!$. This corresponds to 3 628 800 paths for a simple case of 10 nodes. Hence, we use column-generation techniques, where variables are added dynamically.

3 Column generation

In this section, we discuss the various column-generation procedures possible for the decomposed model. In particular, we emphasize the essential fact that different kinds of subproblems are possible.

The column-generation procedure we used is quite standard. A Branch-and-Price (see [BJN+98]) search is executed with the common branching scheme. Integrality constraints on the λ variables are relaxed, and the model solved is restricted to only a limited subset of the variables. Then, as is practical with the revised simplex method, we look for new possible variables entering the restricted model by checking their reduced cost. This operation is known as *pricing* new variables. It uses a subproblem in charge of finding new valid columns with favorable reduced cost. When no more new columns can be priced out, the relaxed solution is optimal for the relaxed problem. If this solution does not contain fractional variables, it is also an optimal solution for the integer problem. Otherwise, branching is done. In each branch, further pricing is needed as the branching rules modify the problem. This Branch-and-Price approach applied to Vehicle Routing Problems is well documented ([Lar99, Koh95, DDS99, CDD+99]).

For several practical reasons, set-partitioning constraints (11) are usually changed to set-covering constraints only. Any solution to the original problem is also solution to this new relaxed problem. Moreover, it is easy to see that an optimal integer solution from the new problem would only contain one route per visit, and hence would be also the optimal solution of the original problem.

3.1 The *pricing* scheme

If π_i and π_0 are the dual values corresponding to equations (11) and (12), the reduced-cost is :

$$rc(p) = c(p) - \sum_{i=1}^{N} a_i^p \pi_i - \pi_0 = \sum_{(i,j)\in p} (d(i,j) - \pi_j) - \pi_0$$

The subproblem thus consists of finding the shortest elementary path with respect to the following cost and respecting the capacity constraint:

$$\begin{array}{rcl} rc(i,j) & = & d(i,j) - \pi_j, \forall j \in \{1,...,N\} \\ rc(i,N+1) & = & -\pi_0 \end{array}$$

3.2 Various possible subproblems

A valid route is an elementary path from depot to depot respecting constraints on some resources (length, capacity, and time). Let p be a path $p = (i_0 = d, i_1, ..., i_m = i, ..., d)$. The resources can be generalized to L resources indexed by l. D_i^l represents the accumulated quantity of ressource l at node i. Distance functions $d^l(i,j)$ define the accumulation of resource l between i and j. Intervals of accepted values $[a_i^l, b_i^l]$ are defined for each pair of resource and visit. We will use the fact that the increase of an accumulated resource between two successive nodes i and j must satisfy the triangle inequality :

$$d^l(i,k) \le d^l(i,j) + d^l(j,k), \ \forall \ l = 1...L$$

The cost of a path is equal to the accumulation of a resource. The reduced cost is obtained from the cost and dual values π_j of each corresponding set-covering constraint as given in the previous section. As only negative reduced-cost routes can enter the restricted master problem, it should be minimized or constrained to be negative.

According to the model, routes should not contain cycles. However most of the algorithms proposed in the literature relax this constraint in order to simplify the subproblem. Indeed, even if the solution of the relaxed restricted master problem may then contain non elementary routes with non-null fractional values, the optimal integer solution will not contain such routes.

Theorem 1 *When the accumulations of resources respect the triangle inequality, an optimal integer solution to the master problem with non-elementary paths contains only elementary paths.*

Proof: If an optimal solution of the integer master problem did contain a non-elementary route, i.e., a vehicle going twice through the same visit, it would be easy to derive a better column from it. Moreover, thanks to the triangle inequality, the route obtained by removing this visit once would always be valid and have an equal or lower cost. A better global solution would hence be feasible; this situation contradicts the hypothesis that the solution containing the non-elementary route was optimal.

In general, the shortest-path subproblems are thus solved by means of a pseudo-polynomial labeling algorithm where cycles are allowed. Note, however, that if only elementary paths are used, the relaxed master problem is more constrained and might offer better quality bounds. We illustrate this result in an example in Section 4.3.

3.3 Labeling algorithm

This kind of algorithm to solve subproblems arising in Vehicle Routing Problems was first introduced in [DD86] and [Des88]. Labeling algorithms associate labels with partial paths and create new ones extending the paths. When no more labels can be generated, the optimal path is given by the label at the end node with the best reduced cost. At some node i, a partial path p is associated with a label $E_p = (rc_p, D_p^0, ..., D_p^L)$. Certain rules are used to fathom partial paths that are known not to be part of the searched optimal path. The basic rule is the *dominance* rule. We say a partial path p_1 ending at node i *dominates* some other partial path p_2 ending at the same node i, if the following conditions are respected:

$$rc_1 \leq rc_2 \tag{13}$$
$$D_1^l \leq D_2^l, \forall l \in \{0, ...L\} \tag{14}$$

Indeed, these conditions ensure that if p_2 could be prolonged to a *good* path p_2^*, then it would be possible to prolong p_1 in the same manner to obtain a path p_1^*. It can be shown that the resulting path p_1^* would be valid and would have a better reduced cost than p_2^*. The partial path p_2 can thus be safely fathomed, and no attempt to prolong it will be made.

In [Des88], a full labeling algorithm for the shortest path problem with resource constraint and time windows is presented. Several implementations are also discussed there.

This dominance rule cannot be used directly to search for the elementary shortest path. This last problem is much more complicated. The central result of this article is to propose a practical, efficient modification of this algorithm for elementary shortest paths.

3.4 Branching scheme

We use the commonly employed branching scheme based on adapted Ryan-Foster rules (e.g. see [DDS92]). We look for two fractional routes sharing the same pair of visits i and j but with one route using the arc (i, j) and not the other. It can be shown that such routes can always be found. In each branch, the shortest-path subproblem can be modified easily (simply by removal of arcs from the underlying graph) without breaking the structure of the problem.

4 Bounds comparison

The treatment of a node of the Branch-and-Price tree ends when no more columns can be priced out. Then the optimal value of the relaxed master problem is a lower bound for the integer problem for the current branch. The quality of those bounds is important, as they make it possible to prune the search tree by fathoming nodes with bad lower bounds. In this section, we present two techniques that can improve the quality of the relaxations. The first one consists of adding cutting planes to the relaxation; it has already been widely documented. The second one is based on an elementary shortest-path subproblem.

When we use SPRCTW (that is, shortest path with resource constraints and time windows), we denote the complete master problem as MP. When we use ESPRCTW (that is, elementary shortest path with resource constraints and time windows), we denote the complete master problem as EMP. When we remove the integrality constraints from MP, we denote it as RMP. When we remove the integrality constraints from EMP, we denote it as REMP.

4.1 Simple example

As an example, let's take a problem with two visits, i and j, very far from the depot d. Let's suppose that:

$$d(d, i) = d(d, j) = 100$$

$$d(i, j) == 1$$

Moreover, let's suppose that capacity and time windows are large. The optimal relaxed solution for EMP is obviously made of the unique route $d-i-j-d$ so that $LB_{elem} = 201$. When cycles are allowed, any legal route of the form $d-(i-j)^n-d$, with $i-j$ repeated n times, (which cost is $200 + 2n - 1$) can be part of a solution with master value $1/n$, providing a bound $LB = 200/n + 2 - \frac{1}{n}$ where n is only limited by the capacity of the vehicles. Such a route is shown in Figure 1.

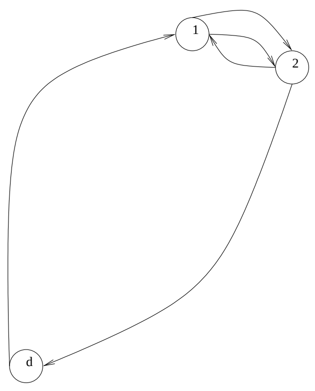

Figure 1: Solution for the VRP example

4.2 Use of cuts

The most widely used method to improve the solution obtained from the relaxed problem is to add cutting planes such as the k-path cuts, also known as subtour cuts when $k = 1$. This family of cuts is discussed in details in [Koh95].

Let S be a set of visits, $\delta(S)$ the set of outgoing arcs from S (that is, $\delta(S) = \{(i, j) \in A / i \in S, j \notin S\}$) and μ_S^p the number of arcs that path p has in $\delta(S)$. With various techniques, it might be possible to demonstrate that the visits from this set cannot be covered with fewer than $\kappa(S)$ vehicles. It is thus possible, for any $k \leq \kappa(S)$ to add a new constraint, known as k-path cut, on the variables λ corresponding to routes containing at least one visit from the set:

$$\sum_{p \in \Omega} \mu_S^p \lambda^p \geq k,$$

It is clear that whichever set S is used, we always have $\kappa(S) \geq 1$. Subtour cuts (i.e. those for $k = 1$) are hence easy to use as no previous result has to be found to prove their validity. The subtour cut applied to our previous example with the set $S = \{i, j\}$ provides a flow of $1/n$ and would then be violated. Adding the cut for this set would improve the lower bound.

However, the bound obtained with MP plus subtour cuts is not equivalent to the bound obtained with EMP. It has been demonstrated in [Koh95] that using ESPRCTW implies that no subtour cuts would be violated. The converse is not true.

Let's illustrate this idea with another example. Suppose we have three visits i, j, and k. The following set of routes can be taken each with the value $\frac{1}{2}$.

$$d - j - k - d$$

$$d - i - j - k - i - d$$

Even if this solution does not violate any subtour cuts, it contains a cycle and hence would not be solution of the EMP.

Finally, we have to remember that even if ESP is harder to solve, when we use cutting planes, we also have to use some other complex algorithm in order to isolate the violated cuts to be added to the relaxation.

4.3 Use of elementary paths

The constraints of MP and EMP being identical and the sets of columns being included one in the other, it is clear that MP is a relaxation of EMP, and RMP is a relaxation of REMP. We saw before that MP and RMP have the same integer solutions. Such a nice result is not true for RMP and REMP, but the relaxation relation implies that the bound obtained with REMP is at least of the same quality as the one obtained with RMP. It might even occur that this tighter relaxation may be integer. Such was the case for almost half the instances we closed using this method. This quality is clearly illustrated in our simple example.

Our idea is thus to use ESPRCTW for the subproblems. This problem being, in theory, much more complex, we have to use an implementation offering a good compromise between worst practical efficiency of the algorithm and better quality of bounds.

5 Elementary shortest path

In this section, we propose several modifications to the labeling algorithm for the elementary path problem.

Two main differences apply:

1. A partial route cannot be continued with a node already present in the partial path.
2. A label cannot be directly fathomed using the usual dominance rule.

The first item means that a partial path ending at i can be prolonged to j only if j has not yet been visited by the partial path. If we denote the set of nodes of the partial path p as $V(p)$, we cannot prolong p to j if we already have $j \in V(p)$.

The second item is the principal difficulty. Indeed, even if some partial path p_1 dominates some other partial path p_2, a continuation of p_2 with some visit $i \in V(p_1)$ might be useful later. Here, we must keep p_2 as p_1 can not be applied the same promising prolongation as in the proof of the dominance rule. In fact, an easily modified dominance rule would be not to fathom any labels and keep them all. Even if this tactic would in theory lead to the optimal elementary path, it is intractable in practice. This practical difficulty seems to be the reason why a relaxation of SPRCTW is accepted.

Our objective is then to improve the dominance rule. The new dominance rule should allow us to fathom as many partial paths as possible with the security that they are not part of the optimal solution, and it must allow us to fathom them as fast as possible.

5.1 A preliminary modified dominance rule

The basic dominance rule cannot ensure that p_2 cannot be continued any better than p_1, because p_2 may perhaps be continued with a node already visited in p_1, but that node cannot be added to p_1. A modified dominance rule (already proposed in [DD86]) is then used where the sufficient condition for a partial path p_1 to dominate another partial path p_2 is to respect the previous conditions (13) and (14), and to have its set of visits $V(p_1)$ included in the set $V(p_2)$.

$$V(p_1) \subseteq V(p_2) \qquad (15)$$

Obviously, this basic new dominance rule will allow many labels to be created. Indeed, when two partial paths are such that the sets of their visits are not included one in the other (and hence condition (15) cannot be respected) then no dominations will be applicable between their prolongations.

We thus introduce different improvements to this rule, both complete and heuristic modifications.

5.2 Exact improvement of the dominance rule

If a partial path p_2 does not contain some visits of another path p_1, the new dominance rule will always keep its label, independently of the value of the resources accumulation, cost, and reduced cost. The reason is simply that the reduced cost of some extension of p_2 could become favorable when going through one of the visits of p_1. This sufficient rule to keep labels is obviously not necessary. It is possible to improve the dominance rule by estimating the expected improvement of reduced cost that could be obtained by going thought some visits of p_1.

Let p_1 and p_2 be two partial paths ending at the same node m. $V(p)$ is the set of nodes visited by the partial path p.

In the case that $V(p_1)\backslash V(p_2) = \emptyset$ and p_1 dominates p_2 according to the original dominance rule, there is no specific problem. This situation shows the new dominance rule in its best case.

Note also that we may not consider some element $n \in V(p_1)\backslash V(p_2)$ in the case where it cannot be appended to p_2 in any way. Indeed, if there is some resource l such that $D_2^l + d^l(m,n) > b_n^l$, it will be impossible to add n. We use here the property of the triangle inequality.

Let's now consider different approximations when $V(p_1)\backslash V(p_2) = \{n\}$:

1. bound on improvement with π_n. Since p_2^* is an extension of partial path p_2, it is possible to create an extension p_1^* of p_1 in the same way, except for n. Using i and j the previous and next node of n in p_2^*, we would thus have:

   $$rc_2^* - rc_1^* = c(i,n) + c(n,j) - c(i,j) - \pi_n + rc_2 - rc_1$$

   Hence p_2^* is better than p_1^* if and only if:

   $$c(i,n) + c(n,j) - c(i,j) - \pi_n + rc_2 - rc_1 < 0$$

   By underestimating the cost part (which is always positive due to the triangle inequality), a sufficient condition to eliminate p_2 is:

   $$-\pi_n \geq rc_1 - rc_2 \tag{16}$$

2. bounds on improvement with $\pi_n - minAddedArcCost_n$. Here, we use a better approximation of the cost part and underestimate it using:

   $$minAddedArcCost_n = min_{i \in prec(n), j \in succ(n)} \left(c(i,n) + c(n,j) - c(i,j)\right)$$

   where $prec(n)$ is the set of possible predecessors, and $succ(n)$ is the set of possible successors of node n. These quantities can be calculated at the initialization of the algorithm for each node n. The sufficient condition is thus:

   $$minAddedArcCost_n - \pi_n \geq rc_1 - rc_2 \tag{17}$$

The two conditions (16) and (17) can then be used to fathom more labels. These conditions can be generalized to the case where $Card(V(p_1)\backslash V(p_2)) > 1$.

However, in practice, it is difficult to find good bounds. We restricted our algorithm to the case where $Card(V(p_1)\backslash V(p_2)) \leq 2$.

5.3 Heuristic improvements of the algorithm

Solving an elementary shortest-path problem with the dominance rule defined in 5.1 is really time-consuming. Even using all the improvements introduced in the previous section, our subproblem might take much time to find only a few new columns. In fact, at the beginning of the treatment of a branch-and-bound node, obtaining a route with negative reduced-cost might not require completeness of the algorithm. We propose in this section some heuristic reductions of the ESPRCTW that can be used in combination with the complete algorithm.

5.3.1 Simple heuristically reduced algorithm

An easy way to obtain a faster heuristic version of our algorithm is to use only item 1 of the differences presented in Section 5 (i.e., it does not allow the extension of a route with any already visited node) but do not modify the dominance rule. Only elementary paths are generated then, but a label can be removed even if the cycle constraint would make it part of the optimal full elementary path. When this algorithm does not return any columns with negative reduced cost, it does not mean that no such column exists, as some partial path might have been erroneously eliminated.

We then use a procedure that consists of relying on the heuristic algorithm as long as it is able to provide new negative reduced-cost routes, and when that is no longer possible, we use the complete version. When the complete version also fails to find a valid new column, the treatment of the node is finished.

5.3.2 Dominance levels

There is a noticeable gap between the two versions of the algorithm used in the previous procedure. Sometimes, the heuristic version quickly ends without returning any new columns, but the execution of the complete version would still be very time consuming. We introduced a parameter to try to control the *correctness* of the algorithm almost continuously. The parameter *DominanceLevel* defines the length (i.e. number of nodes) of partial paths after which the complete new dominance rule is applied. If the partial path is shorter, the heuristic rule is applied. Then the two versions of the algorithm (heuristic and complete) defined before correspond respectively to the two extreme values ∞ and 0 of *DominanceLevel*. The procedure now consists of starting with a huge value for *DominanceLevel*, and then decreasing it when no new column can be found.

5.3.3 Graph reductions

As proposed in other publications for the problem of a shortest path with cycles (e.g. [DDS99]), the visit graph can be heuristically reduced in order to accelerate the algorithm. We eliminate arcs with respect to their *distance* for some resource.

Different criteria have been used, such as:

$$d^l(i,j) \geq percent * max_{k \in succ(i)} d^l(i,k)$$
$$d^l(i,j) \geq (1 + percent) * min_{k \in succ(i)} d^l(i,k)$$

As in previous reductions, we increase the percentage of graph kept when no new path is found.

We note that the three procedures previously defined are globally complete (i.e., no column is forgotten), even if at some point they use a heuristic version of the elementary shortest-path algorithm, as the full version is run in case of failure.

5.4 Implementation-oriented improvements

The previous subsection introduced some algorithmic improvement to the global pricing procedure for an elementary shortest path. We present here some ideas that we used to implement it in an efficient manner.

5.4.1 Storage of labels

As we use many successive runs of slightly modified and limited versions of the algorithm, much time could be used in the *warm up* of the algorithm, i.e., in the re-creation of many labels already present at the end of previous run. We thus used storage of labels as an incremental tool. This storage can be filled with labels from any of the algorithms used, and then used to fill in a new algorithm with the actual labels before it is run.

5.5 Other improvements

Several other improvements have been used that had yet been proposed before :

- Time-window based graph reduction :

  As proposed in [DD86], we removed from the set of possible arcs those made impossible for reasons based on time windows. If the condition:

  $$a_i^l + d^l(i,j) > b_j^l$$

  is satisfied for a resource l, then the arc (i,j) cannot be part of any solution path. It can be removed from the initial graph.

- Minimal number of paths :

  As proposed in [Lar99], we stop the execution of the algorithm when a fixed minimal number of paths and labels have been generated. This reduction is particularly interesting as when no path is found, we know such a path does not exist, and we do not have to rerun another version of the algorithm.

- Reduced-cost based graph reduction :

  Finally, as proposed in [RGP02], we also reduce the initial graph, taking into account reasons based on reduced cost.

Instances	nbVisits	$\frac{LB-LB(1)}{LB(1)}$	$\frac{LB(2)-LB(1)}{LB(1)}$	$\frac{LB-LB(1)}{OPT-LB(1)}$	$\frac{LB(2)-LB(1)}{OPT-LB(1)}$
R1	50	0,38%	0,33%	23,56%	36,86%
R1	100	0,32%	0,15%	13,95%	15,59%
RC1	50	0,55%	8,89%	11,62%	53,03%
RC1	100	0,55%	1,82%	3,06%	38,77%

Table 1: Lower bounds for series 1

6 Computational results

6.1 Solomon's benchmark

We worked on the Solomon instances [Sol87], which are widely used both for exact and heuristic methods applied to the VRPTW. We followed the common conventions used for exact methods: minimizing only the distance, distances and durations use Euclidian distances between pairs of coordinates rounded to first lower decimal. Two series of instances exist, and the first series has smaller time windows. This series is thus easier to solve, as more constrained. Many studies concentrate their effort on this series. We give results mainly about the second series. We focused our interest there as it contains most of the still open instances. Instances are also divided in three classes: class "C" in which visits are clustered, class "R" where visits are randomly distributed, and class "RC" with mixed distributions. Each instance is a problem of 100 visits from which are extracted two smaller problems using the first 25 and 50 visits.

6.2 Bounds

When no more new columns can be priced out, a solution of the relaxed restricted master problem is a lower bound for the current Branch-and-Price branch.

In the case of the root node, we obtain a global lower bound for the complete problem. When working on a subproblem with only elementary paths, we solve a more constrained problem, and the bounds obtained are theoretically better. In Table 1, we compare the bounds obtained on series R1 and RC1 with those given in [CR99]. The two comparisons given are:

- between our lower bound LB and $LB(1)$ from [CR99], corresponding to to a subproblem forbidding only cycles with the form $i - j - i$.
- between $LB(2)$ and $LB(1)$ from [CR99]. $LB(2)$ corresponds to $LB(1)$, plus the addition of subtour and 2-path cuts at the root node.

We can see in this table that our lower bound is indeed better than the one obtained with non-elementary paths. On average, the gap between our lower bound and the optimal solution is 17% smaller than the gap between $LB(1)$ and the optimal solution. The same calculation would give 47% with $LB(2)$ and suggests that the use of cutting planes could be added to our algorithm.

At other nodes, lower bounds allow us to prune the search tree and reduce the total time needed to find and prove optimal solutions. This consequence, among others, allowed us to close some open instances. Table 2 provides more details about the improvements of lower bounds obtained for some instances of series 2.

The bold lines correspond to the instances we closed. Our improved bounds for most of the RC2 instances we closed correspond to an integral solution, and no branching is needed.

Instance	nbVisits	LB	LB(1)	OPT	$\frac{LB-LB(1)}{LB(1)}$	$\frac{LB-LB(1)}{OPT-LB(1)}$
R204	**25**	**350,47**	**337,025**	**355,0**	**3,99%**	**74,80%**
R208	**25**	**328,20**	**321,752**	**328,2**	**2,00%**	**100,0%**
R201	50	791,90	788,625	791,9	0,42%	100,00%
R202	50	698,50	694,162	698,5	0,62%	100,00%
R203	**50**	**598,58**	**592,383**	**605,3**	**1,05%**	**48,00%**
R205	**50**	**682,85**	**668,524**	**690,1**	**2,14%**	**66,40%**
R206	**50**	**626,34**	**613,445**	**632,4**	**2,10%**	**68,05%**
R209	**50**	**599,83**	**585,939**	**600,6**	**2,37%**	**94,75%**
R210	**50**	**636,10**	**627,385**	**645,6**	**1,39%**	**47,85%**
R201	100	1140,30	1139,746	1143,2	0,05%	16,04%
RC203	**25**	**326,90**	**220,182**	**326,9**	**48,47%**	**100,0%**
RC204	**25**	**299,70**	**191,221**	**299,7**	**56,73%**	**100,0%**
RC208	**25**	**269,10**	**163,009**	**269,1**	**65,08%**	**100,0%**
RC201	50	684,80	678,867	684,8	0,87%	100,0%
RC202	**50**	**613,60**	**516,619**	**613,6**	**18,77%**	**100,0%**
RC203	**50**	**555,30**	**421,146**	**555,3**	**31,85%**	**100,0%**
RC205	**50**	**630,20**	**567,970**	**630,2**	**10,96%**	**100,0%**
RC206	**50**	**610,00**	**447,305**	**610,0**	**36,37%**	**100,0%**
RC207	**50**	**558,60**	**395,725**	**558,6**	**41,16%**	**100,0%**
RC201	100	1255,94	1244,456	1261,8	0,92%	66,22%
RC202	**100**	**1088,08**	**1012,616**	**1092,3**	**7,45%**	**94,70%**
RC205	**100**	**1147,61**	**1063,922**	**1154,0**	**7,87%**	**92,90%**

Table 2: Lower bounds for some series 2 instances

6.3 Optimality: new instances closed

Table 3 gives our results for 17 Solomon instances previously open for which we found and proved optimal solutions. Also given: the value of the optimal solution (*OPT* column), the number of vehicles used, the relative difference between lower bound at the root node and optimal solution (*gap* column), the number of explored nodes, the number of solved subproblems (*SP* column), the time to find the optimal solution and the time to prove the optimality.

The given timings have been obtained using a 1,5 Ghz Pentium IV with 256 Mo and using ILOG JNI CPLEX 7.5, ILOG JSolver 1.0 and the JVM from IBM for Linux version 1.3.0.

The instances noted with (*) and (**) are those for which we found a better optimal solution that those given by [Lar99] et [KLM01], respectively. On those instance, our results have been validated by Jesper Larsen, indicating theirs are hence not optimal.

Instance	OPT	V	Gap (%)	N	SP	T_{opt}	T_{proof}
R203.50	605,3	5	1,11	15	432	117,8	3320,9
R204.25	355,0	2	1,27	17	365	88,4	171,6
R205.50**	690,1	4	1,05	95	1551	301,4	531,0
R206.50	632,4	4	0,96	67	1319	2972,8	4656,1
R208.25*	328,2	1	0,00	1	64	705,7	741,5
R209.50	600,6	4	0,13	5	144	120,8	195,4
R210.50	645,6	4	1,47	977	16183	25151,1	65638,6
RC202.50	613,6	5	0,00	1	70	5,3	13,0
RC202.100	1092,3	8	0,39	39	1861	18053,8	19636,5
RC203.25*	326,9	3	0,00	1	65	4,0	5,1
RC203.50	555,3	4	0,00	1	315	4479,4	4481,5
RC204.25	299,7	3	0,00	1	58	1,95	13,0
RC205.50*	630,2	5	0,00	1	82	10,3	10,6
RC205.100	1154,0	7	0,55	71	2706	3131,6	15151,7
RC206.50	610,0	5	0,00	1	61	8,6	9,4
RC207.50	558,6	4	0,00	1	107	66,0	71,1
RC208.25	269,1	2	0,00	1	185	32239,3	33785,3

Table 3: The Solomon instances we closed.

Conclusion

In this article, we returned to the Dantzig-Wolfe decomposition of the Vehicle Routing Problem. The elementary shortest-path subproblem involved in the resulting column generation has been, as far as we know, always relaxed to a shortest-path problem where more efficient algorithms are available. We propose several improvements to the usual elementary shortest-path algorithm. We also show that the benefits of using elementary shortest-path might be practically useful when this modified algorithm is used. Indeed, better bounds are found, and new instances can be closed.

As already mentioned, another way of improving the lower bounds which led us to find new optimal solutions is to use cutting planes. A logical continuation of this work would be to combine the two proposals. Also, we provide results in [CDP02] about how our work can cooperate with other techniques such as local search.

7 Acknowledgements

This work is partially the result of wider investigations on collaborative algorithms for Vehicle Routing Problems done in collaboration with Emilie Danna and Claude Le Pape. Many interesting discussions with Javier Lafuente also contributed to the development of the ideas presented here. We also thank Jesper Larsen, who validated the corrections found on some optimal solutions, and Kathleen Callaway for the proofreading.

References

[BH01] R. Bent and P. Van Hentenryck, *A two-stage hybrid local search for the vehicle routing problem with time windows*, Tech. Report CS-01-06, Brown University, septembre 2001.

[BJN+98] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance, *Branch-and-price: column generation for solving huge integer programs*, Operations Research **46** (1998), 316–329.

[CDD+99] J.F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Salomon, and F. Soumis, *The vrp with time windows*, Les Cahiers du GERAD (1999), no. G-99-13.

[CDP02] A. Chabrier, É. Danna, and C. Le Pape, *Coopération entre génération de colonnes et recherche locale appliquée au problème de routage de véhicules*, JNPC, 2002.

[CLM01] J.-F. Cordeau, G. Laporte, and A. Mercier, *A unified tabu search heuristic for vehicle routing problems with time windows*, Journal of the Operational Research Society **52** (2001), 928–936.

[CR99] W. Cook and J.L. Rich, *A parallel cutting-plane algorithm for the vehicle routing problem with time windows*, Tech. Report TR99-04, Department of Computational and Applied Mathematics, Rice University, May 1999.

[DD86] Y. Dumas and J. Desrosier, *A shortest path problem for vehicle routing with pick-up, delivery, and time windows*, Les Cahiers du GERAD (1986), no. G-86-09.

[DDS92] M. Desrochers, J. Desrosiers, and M. Solomon, *A new optimization algorithm for the vehicle routing problem with time windows*, Operations Research **40** (1992), 342–354.

[DDS99] G. Desaulniers, J. Desrosiers, and M. M. Salomon, *Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems*, Les Cahiers du GERAD (1999), no. G-99-36.

[Des88] M. Desrochers, *An algorithm for the shortest path problem with resource constraints*, Les cahiers du GERAD (1988), no. G-88-27.

[DFS+00] B. De Backer, V. Furnon, P. Shaw, Ph. Kilby, and P. Prosser, *Solving vehicle routing problems using constraint programming and metaheuristics*, Journal of Heuristics **6** (2000), 501–523.

[Dro94] M. Dror, *Note on the complexity of the shortest path models for column generation in vrptw*, Operations Research **42** (1994), 977–978.

[DW60] G.B. Dantzig and P. Wolfe, *Decomposition principle for linear programs*, Operations Research **8** (1960), 101–111.

[FDGG02] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen, *An exact algorithm for the elementary shortest path problem with resource constraints : Application to some vehicle routing problems*, 2002.

[GTA99] L.M. Gambardella, É. Taillard, and G. Agazzi, *MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows*, New Ideas in Optimization (David Corne, Marco Dorigo, and Fred Glover, eds.), McGraw-Hill, London, 1999, pp. 63–76.

[HG99] J. Homberger and H. Gehring, *Two evolutionary metaheuristics for the vehicle routing problem with time windows*, INFOR **37** (1999), 297–318.

[KLM01] B. Kallehauge, J. Larsen, and O.B.G. Madsen, *Lagrangean duality and non-differentiable optimization applied on routing with time windows - experimental results*, Tech. Report IMM-TR-2001-9, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, August 2001.

[Koh95] N. Kohl, *Exact methods for time constraints routing and scheduling problems*, Ph.D. Thesis, Institute of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark (1995).

[KPS00] Ph. Kilby, P. Prosser, and P. Shaw, *A comparison of traditional and constraint-based heuristic methods on vehicle routing problems with side constraints*, Constraints **5** (2000), no. 4, 389–414.

[Lar99] J. Larsen, *Vehicle routing with time windows - finding optimal solutions efficiently*, http://citeseer.nj.nec.com/larsen99vehicle.html, 1999.

[RGP99] L.-M. Rousseau, M. Gendrau, and G. Pesant, *Using constraint-based operators in a variable neighborhood search framework to solve the vehicle routing problem with time windows*, CP-AI-OR (1999).

[RGP02] L.-M. Rousseau, M. Gendreau, and G. Pesant, *Solving small VRPTWs with constraint programming based column generation*, CP-AI-OR, 2002, pp. 333–344.

[RT95] Y. Rochat and E. Taillard, *Probabilistic diversification and intensification in local search for vehicle routing*, Journal of Heuristics **1** (1995), 147–167.

[Sha98] P. Shaw, *Using constraint programming and local search methods to solve vehicle routing problems*, Principles and Practice of Constraint Programming, 1998, pp. 417–431.

[Sol87] M.M. Solomon, *Algorithms for the vehicle routing and scheduling problem with time window constraints*, Operations Research **35** (1987), 254–265.