Arcs-States models for the Vehicle Routing Problems: new improvement methods

T. Mautor, E. Naudin

PRiSM - Université de Versailles Saint-Quentin en Yvelines
45, avenue des États-Unis
78035 Versailles Cedex
Thierry.Mautor@prism.uvsq.fr, Edith.Naudin@prism.uvsq.fr

ABSTRACT

We present in this paper different Arcs-States models for the Vehicle Routing Problem with cumulative resource constraints. In these models, the variables correspond to the states (i.e. the quantities of resource) in which are the vehicles when they use an arc. The LP relaxation of the problem provides a lower bound that can be embedded in a Branch and Bound algorithm that solves exactly the problem. Our models contain a pseudo-polynomial number of variables and constraints and column and row generations have to be used. Generally, in a Branch and Bound algorithm, the lower bound needs to be very effective in order to strongly reduce the size of the Branch and Bound trees. However, one of the models we present, called Dates-Only, lies on a relaxation of the Arcs-States model where a resource is removed from the states in the variables. Although the quality of the bounds decreases, the global resolution time is widely improved, as it can be illustrated on instances of Solomon's benchmark.

KEYWORDS Branch & Bound method, column and row generation, vehicle routing problem

1 Introduction

The Vehicle Routing Problems (VRP) involves the design of a set of routes for a fleet of vehicles, starting and ending at a depot and which services a set of customers with known demands. Each customer has to be visited by one of these routes and the objective is to minimize the global cost of the routes.

The best exact methods usually use a modelling of the problem where the variables represent the different feasible routes. By extension, this model will be called in this paper “the Routes model”. Such a model has been first proposed for the VRP with Time Windows by Desrosiers, Soumis and Desrochers, [DSD84]. Due to the huge number of variables of this model, the LP relaxation is generally solved by column generation. The subproblem searches (dynamic programming procedure) the feasible routes of minimal marginal cost which are progressively added to the master problem.

We have proposed a new model in which the variables correspond to the
state of a vehicle (in terms of load, time, ...) when it uses an arc. In this paper, we briefly present this model and the corresponding resolution scheme by column and row generation. We also quickly describe additional techniques, proposed for the model Routes but that can be adapted to our model, such as the elimination of the 2-cycles or the 2-path cuts. But we specially focus on a relaxation of the Arcs-States model where a resource is removed from the states in the variables. Although the quality of the bounds decreases, their generation is widely accelerated and the global resolution is more efficient. We illustrate these results on some instances of the classical benchmark of Solomon.

2 The Arcs-States model

2.1 The Vehicle Routing Problem with Cumulative constraints (VRPC)

There are many alternatives of Vehicle Routing Problems with a wide range of possible constraints and with variations in the cost function, in the number of depots, in the variety of vehicles, ... We choose to present here a rather general version of the VRP with :

- a fleet of identical vehicles in unlimited number,
- a single starting base for the vehicles noted 0 and a single depot of arrival noted \( n + 1 \) where \( n \) is the number of customers,
- a set of customers noted \( C \),
- unspecified cumulative resource constraints - more precisely, all the constraints are modelised in the same form : a cumulative resource which progressively increases in function of the traversed arcs and which must remain, at each visited customer, in a given interval. We consider a set \( L \) of resources as well as a consumption \( d_{l}^{i_j} \) of each resource \( l \in L \) for each arc \((i, j)\). For each resource \( l \) and each customer \( i \), the value of the resource \( l \) must be in the interval \([a_{l}^{i}, b_{l}^{i}]\) at the arrival at the customer \( i \)
- some costs entirely assigned to the arcs that are noted \( c_{i_j} \).

2.2 Arcs-States model

The variables used in the Arcs-States model are written under the following form : \( x_{e}^{i_j} \), where \((i, j)\) is an arc and \( e \) a vector of states. The variable \( x_{e}^{i_j} \) is equal to 1 if the resources of the vehicle that leaves the node \( i \) are described by the vector \( e \). More precisely, \( e = (e^1, \ldots, e^{L_l}) \), with \( e^l \) representing the state of the vehicle that traverses the arc \((i, j)\) according to the resource \( l \in L \). The following notations are used in this model :

- the sets \( \Gamma^+ \) et \( \Gamma^- \) respectively correspond to the successors and to the predecessors of a node,
- the sets \( \Lambda \) contain all the feasible vectors of states for a given customer :
  \( \forall i \in C, \Lambda_i = \{ e \in \mathbb{N}^{L_l} / \forall l \in L, a_l^i \leq e^l \leq b_l^i \} \),
- the operator \( \oplus_{i_j} \) calculates the vector of states after the use of arc \((i, j)\) :
  \( \forall i \in C, \forall j \in C, \forall e_i \in \Lambda_i, e_i \oplus_{i_j} = e_j \iff \forall l \in L, e_j^{l} = \max(a_l^i + d_{l}^{i_j}, c_{i_j}) \).
• the sets $\Delta$ contain all the feasible vectors of states for a given arc:
  $\forall i \in C, \forall j \in C, \Delta_{ij} = \{e \in \Lambda_i / a_i \leq e \leq b_i, a_j \leq e \leq b_j \}$.

The model $(\text{VRPC}_{AS})$ can be written under the following form:

$$
\begin{align*}
\min & \quad \sum_{i \in C} \sum_{j \in C} \sum_{e \in \Delta_{ij}} c_{ij} x_{ij}^e \\
\text{s.t.} & \quad \sum_{j \in C} \sum_{e \in \Delta_{ij}} x_{ij}^e \leq n, \quad \forall i \in C \\
& \quad \sum_{j \in C} x_{ij}^e \geq 1, \quad \forall i \in C \\
& \quad \sum_{h \in \Gamma_i^+} \sum_{e \in \Delta_{hi}} x_{hi}^e - \sum_{j \in C} \sum_{e \in \Delta_{ij}} x_{ij}^e = 0, \quad \forall i \in C \\
& \quad \sum_{h \in \Gamma_i^-} \sum_{e \in \Delta_{hi}} x_{hi}^e - \sum_{j \in C} \sum_{e \in \Delta_{ij}} x_{ij}^e \geq 0, \quad \forall i \in C, \forall e \in \Lambda_i \\
& \quad x_{ij}^e \in \{0; 1\} \quad \forall i \in C, \forall j \in C, \forall e \in \Delta_{ij}
\end{align*}
$$

The constraints (2) impose a limit to the number of vehicles used. The constraints (3) impose that each customer is served by at least one vehicle. This classical relaxation consists in considering a Set Covering Problem rather than a Set Partitioning Problem. The constraints (4) ensure the flow conservation at each node.

The set of constraints (5) verify the compatibility of the vectors of states arriving at a given customer with the vectors of states leaving this customer. For instance, on a single resource (the time), the constraint

$$
\sum_{h \in \Gamma_i^-} \sum_{e \in \Delta_{hi}} x_{hi}^e - \sum_{j \in C} \sum_{e \in \Delta_{ij}} x_{ij}^e \geq 0, \quad \forall i \in C, \forall e \in \Lambda_i
$$

date $e$ and for any customer $i$, the flow that arrives at this customer before $e$ is greater than the flow that leaves $i$ before this date $e$. In others words, a vehicle may wait at a customer. More generally, this relaxation allows a quantity of resource (load, time, ...) to be lost at a customer. The number of variables and of constraints of this model is proportional to the number of possible vectors of states which is pseudo-polynomial, it depends on the size of the time windows, the capacity of the vehicles, ...

### 2.3 Exact resolution of an Arcs-States model

The LP relaxation of our model $\text{VRPC}_{AS}$ (relaxation of the constraints (6)) provides a lower bound that can be embedded in a Branch and Bound algorithm for an exact solution of the problem.

However, the large number of variables and constraints leads to solve this LP relaxation by column and row generations. The algorithm starts with an initial problem limited in terms of variables and constraints. Thus, in the initial limited problem, all the constraints (5) are omitted. Variables and constraints are progressively added until the problem obtained is both feasible and optimal.
with respect to the global problem. The variables are selected by their negative marginal cost and the selected constraints are the ones that do not belong to the current problem and that are violated in the current optimal solution.

**Column generation algorithm:** In order to quickly generate a subset of good variables, we consider the arcs-states variables of the minimal marginal cost routes. This subproblem consists in finding a constrained shortest path in the graph where the costs on the arcs are $\hat{c}_{ij} = c_{ij} - \alpha_i$, where $\alpha_i$ is the dual value associated to the constraint (3) ($\alpha_0$ for the constraint (2)). Thus, dynamic programming algorithms can be used (see [DS88]).

**Row generation algorithm:** The dual subproblem consists in detecting the constraints (5) both omitted in the current problem and violated in the current solution. We have shown that this problem is equivalent to a maximal flow problem. Indeed, the constraints (5) verify, for each customer, the compatibility of the vectors of states of the current solution.

It is thus possible to build, for each customer $i$, a bipartite graph in which the nodes of the first set correspond to the arcs-states arriving at $i$ while the nodes of the second set are the arcs-states leaving $i$. An arc is created between a node $n_1$ of the first set and a node $n_2$ of the second set when each component of the vector of states associated to $n_1$ is lower than the corresponding component of the vector of states associated to $n_2$. The flow passing by each node is bounded by the value of the arc-state variable associated to this node. No constraints of type (5) is violated for the customer $i$ if and only if the value of the maximal flow in this bipartite graph is equal to 1. Otherwise, a marking procedure allows to identify the sets of incompatible nodes in the current solution and to generate the corresponding constraint that is violated.

**3 First improvement methods**

Our first computational experiments have shown that the Arcs-States model presented in the previous section provides some good results but is not completely competitive with the best exact methods for the VRPs based on “the model Routes” where numerous additional techniques have been proposed through the years that have substantially improved the performances.

That is the reason why, in a first time, we have adapted some of these improvement methods to our model. We briefly describe these adaptations in this section. Mainly, these methods have been developed for the Vehicle Routing Problem with Capacity and Time Windows constraints (VRPCTW).

One well known improvement has been proposed by Desrochers, Desrosiers et Solomon in 1992 [DDS92] and consists in eliminating the 2-cycles in the generation of the routes by the dynamic programming algorithm. So, the quality of the lower bound is enhanced.

For the adaptation of this technique to the Arcs-States model, we need to consider variables $x_{ij}^{h,c}$ where $h$ is the node visited before the arc $(i, j)$. Obviously, such a variable is valid only if $h \neq j$. But, the use of these variables leads also
to modify the constraints of flow conservation at a node (constraints (4)) and the constraints of flow conservation at a node-states (constraints (5)).

Another technique has been proposed by Kohl et al., [KDMSS99] in order to obtain better lower bounds: the 2-path cut. More generally, the \( k \)-path cuts forbid the solutions where an insufficient number of vehicles serve a subset of customers identified as requiring at least \( k \) vehicles. If \( S \) is a set of customers, let us note \( x(S) \) the flow entering in \( S \) and \( k(S) \) the minimal number of vehicles needed to satisfy \( S \). Then \( x(S) \geq k(S) \) is a valid inequality. The difficulty of finding \( k(S) \) leads to consider only the \( k \)-path cuts with \( k \leq 2 \). It is easy to adapt these cuts to the Arcs-States model since they are formulated with variables “arc”. To use these cuts, we do not need to modify our model.

To reduce the number of nodes in the Branch & Bound tree, we classically use an upper bound. The higher the quality of this upper bound is, the more we can cut some branches in the tree. This is the reason why we compute, with a Mixed Integer Programming solver (Cplex), the optimal integer solution of the partial problem (some variables and some constraints are missing) which is generated at the root of the Branch & Bound.

### 4 The Dates-Only model

The trend of the improvement methods described in the previous section, used for the “model Routes” and we adapted to our Arcs-States model is rather to enhance the quality of the lower bounds by the addition of new cuts. At this opposite of this trend, we tried to reduce the computation time necessary to find the routes of minimal marginal cost. For that, we proposed another model Arcs-States in which a resource is removed from the states in the variables. It can thus be considered as a relaxation of the Arcs-States model. But, the limitation in this resource does not appear anymore in the set of feasible states for the variables and needs to be expressed by some constraints.

For instance, for the VRPCTW, we have two resource constraints: time and capacity. If we choose to keep the time dimension in the arcs-states variables and to remove the capacity dimension, the variables become \( x_{ij}^{(d)} \) (\( d \) for the date) instead of \( x_{ij}^{(q,d)} \) (\( q \) for the load). But some constraints need to be added to limit the load of the vehicles.

#### 4.1 The capacity constraints

As the arcs-states variables do not contain the vehicle indices (vehicles are identical), it is not possible to express the capacity constraints under a simple form as we could do otherwise: for instance \( \sum_{i,j,d} \text{dem}_i \times x_{ij}^{v,d} \leq Q, \forall v \), where \( v \) is the indice of the vehicle.

Consequently, we have to identify the routes for which the total load is largest than the capacity of the vehicles and to forbid these routes.

We illustrate our constraints with the following example (figure 1). The left
part of the figure 1 gives an example of road where the sum of demands is 240 whereas the limit is 200. A constraint is thus generated to forbid this road. With this additional constraint, the solution may become for example the one presented in the right part of the figure 1. In the new partial solution, the edge (6;7) is not used and the initial road is thus cut into two new roads. This second solution is feasible.

\[ x_{1;2} + x_{2;3} + x_{3;4} + x_{4;5} + x_{5;6} + x_{6;7} + x_{7;8} + x_{8;9} + x_{9;n+1} \leq 8. \]

**Figure 1.** Unsatisfied capacity constraint

A route \( R = (0, \sigma_1, \sigma_2, \ldots, \sigma_l, n + 1) \) can be forbidden with the constraint \( \sum_{p=1}^{l-1} x_{\sigma_p,\sigma_{p+1}} \leq l - 1 \) which limits the number of arcs used (we use the notation \( x_{ij} = \sum_{d \in \Delta_{ij}} x_{ij}^d \)). If the demand of customer \( i \) is noted \( dem_i \) and \( Q \) represents the vehicle’s capacity, then for each route \( R \) with a total demand \( \sum_{p=1}^{l} dem_{\sigma_p} \) greater than \( Q \), the constraint \( \sum_{p=1}^{l} x_{\sigma_p,\sigma_{p+1}} \leq l - 1 \) forbids this route. Such a constraint is valid and can be considered in the problem. As an illustration, on the instance given on figure 1, this constraint is

\[ 40 \times x_{1;2} + 10 \times x_{2;3} + 20 \times x_{3;4} + 40 \times x_{4;5} + 40 \times x_{5;6} + 30 \times x_{7;8} + 20 \times x_{8;9} + 10 \times x_{9;n+1} \leq 200 \]

is not satisfied (210 > 200) while the solution is feasible.

### 4.2 The model

So in this model, called (VRPCTWDO), the variables are changed in variables \( x_{ij}^d \) and a set of linear capacity constraints is added.

Of course, in this model, the number of variables remains pseudo-polynomial. In comparison with the previous model, the column generation algorithm and the corresponding dynamic programming algorithm for the search of constrained shortest path are markedly accelerated by the relaxation of the capacity constraints.

Moreover, the number of flow-states conservation constraints (constraints (10)) is widely reduced by the non-consideration of the load. On the other side, the size of the set of capacity constraints (constraints (11)) is exponential and, consequently, these constraints are generated only when they are not satisfied by the current solution.

### 5 Computational tests and results

We present the results of four versions:
\[
\begin{align*}
\text{min} & \quad \sum_{i \in C} \sum_{j \in C} \sum_{d \in \Delta_{ij}} c_{ij} x_{ij}^d \\
\text{s.t.} & \quad \sum_{j \in C} \sum_{d \in \Delta_{ij}} x_{ij}^d \leq n, \quad \forall i \in C \\
& \quad \sum_{j \in C} \sum_{d \in \Delta_{ij}} x_{ij}^d \geq 1, \quad \forall i \in C \\
& \quad \sum_{h \in \Gamma_i} \sum_{d \in \Delta_{hi}} x_{hi}^d = \sum_{j \in C} \sum_{d \in \Delta_{ij}} x_{ij}^d = 0, \quad \forall i \in C \\
& \quad \sum_{p=1}^{\frac{|S|}{2}} \sum_{d \in \Delta_{\sigma_p \sigma_{p+1}}} x_{\sigma_p \sigma_{p+1}}^d \leq |S| - 1 \quad \forall S = \{\sigma_1, \ldots, \sigma_l\} \subseteq C \\
& \quad \sum_{h \in \Gamma_i} \sum_{d \in \Delta_{hi}} x_{hi}^d \leq \sum_{j \in C} \sum_{d \in \Delta_{ij}} x_{ij}^d \geq 0, \quad \forall i \in C, \forall d \in \Delta_{ij} \\
& \quad x_{ij}^d \in \{0; 1\} \quad \forall i \in C, \forall j \in C, \forall d \in \Delta_{ij}
\end{align*}
\]
<table>
<thead>
<tr>
<th>Version Ω</th>
<th>A</th>
<th>B₁</th>
<th>B₂</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nb. of instances solved ((N I_Ω))</td>
<td>43</td>
<td>49</td>
<td>41</td>
<td>57</td>
</tr>
<tr>
<td>Average computational time ((T_Ω)) (sec.)</td>
<td>103.50</td>
<td>20.21</td>
<td>248.32</td>
<td>13.99</td>
</tr>
<tr>
<td>Average number of nodes ((N N_Ω))</td>
<td>101.89</td>
<td>57.57</td>
<td>368.38</td>
<td>60.32</td>
</tr>
<tr>
<td>Avg. ratio LB - optimum ((LB_Ω/OPT))</td>
<td>99.06 %</td>
<td>99.60 %</td>
<td>98.74 %</td>
<td>99.54 %</td>
</tr>
<tr>
<td>Avg. time to compute one node ((TN_Ω))</td>
<td>2.59 sec.</td>
<td>10.15 sec.</td>
<td>0.25 sec.</td>
<td>1.37 sec.</td>
</tr>
</tbody>
</table>

LB is the lower bound of the root. The difference between \(B_1\) and \(B_2\) that we mentioned above on the quality of bounds, size of trees and time to compute a bound is clearly confirmed by this table.

But, above all, the version C is clearly better than the other versions. All the disadvantages of versions \(B_1\) and \(B_2\) are reduced in the version C. The average time to compute one node is widely reduced in comparison to \(B_1\) while the size of the trees is widely reduced in comparison to \(B_2\). Moreover, the number of instances exactly solved is increased with this version.

In order to detail this analysis, we give a second table in which we compute the impact of adding an option. For example, in a column \(Ω_0 \rightarrow Ω_1\), we give respectively the values on the rows \(\frac{N I_{Ω_0}}{N I_{Ω_1}}, \frac{T_{Ω_0}}{T_{Ω_1}}, \frac{N N_{Ω_0}}{N N_{Ω_1}}, \frac{LB_{Ω_0} - LB_{Ω_1}}{OPT - LB_{Ω_0}}\) and \(\frac{TN_{Ω_0}}{TN_{Ω_1}}\).

\[
\begin{array}{cccccc}
\text{Ratio number of instances solved} & A \rightarrow B_1 & B_1 \rightarrow C & A \rightarrow B_2 & B_2 \rightarrow C & A \rightarrow C \\
0.20 & 0.69 & 0.39 & 1.39 & 1.34 \\
0.56 & 1.05 & 3.62 & 0.16 & 0.59 \\
0.58 & -0.16 & -0.34 & 0.64 & 0.51 \\
3.92 & 0.14 & 0.10 & 5.44 & 0.53 \\
\end{array}
\]

### 5.1 The Dates-Only model

Our first analysis consists in detailing the behaviour and the contribution of the Dates-Only model (version \(B_1 \rightarrow C\) and version \(A \rightarrow B_2\)).

- The first remark is not very positive but the single use of the Dates-Only model is not very good (\(A \rightarrow B_2\)). The average computational time is significantly higher, the quality of the bound is clearly lower and eventually the number of instances solved decreases slightly.
- But, this negative effect of the Dates-Only model completely disappears when the options 2-cycles elimination, 2-path cuts and computation of an upper bound are used (\(B_1 \rightarrow C\)). The quality of the lower bound does not significantly decreases and thus the number of nodes very slightly increases (ratio 1,05).
- The number of instances solved is very clearly increased when the Dates-Only model is used in addition of the other options. In the same way, the average computational time is widely reduced on our population sample.

### 5.2 Options 2-cycle elimination, 2-path cuts and upper bound

Our second analysis goes deeper into the behaviour and the contribution of the different options we have adapted to our model (version \(B_2 \rightarrow C\) and version \(A \rightarrow B_1\)).
• First, the use of these options in an Arcs-States model is always positive. The column $A \rightarrow B_1$ shows the number of instances solved and the average times are better when using the different options to improve the bounds.

• The use of the Dates-Only model in addition to these options is very good in the measure where the negative effect on the quality of the bounds and on the size of the Branch & Bound trees is very limited. So, the model can fully benefit from the reduction in the time for the lower bound and this explains that the average time of resolution and the number of instances solved are clearly better when the Dates-Only model is used.

Finally, the three options that improve the quality of the bounds give some very good results but these results are even more improved by the Dates-Only model.

6 Conclusion

In order to improve the performances of the original model we have proposed: the Arcs-States model, we have proposed and tested two kinds of approaches. On one hand, we have adapted to our model a set of methods: 2-cycles elimination, 2-path cuts and computation of upper bounds which enhance the quality of the bounds. With this approach, the number of nodes is reduced but the lower bound computation of a node is slowed down. On the other hand, the Dates-Only model that we propose accelerates the lower bound computation, one dimension being removed in the dimension of the vectors of states, but the quality of the bound decreases and the trees are larger. The computational results show that it is when these two kinds of approaches are mixed that the results are the best. The weaknesses of each approach are offset by the other approach. Finally, the version that uses all the options, has an acceleration equal to 0.14 (7 times faster) on our population sample and solves 33% of additional instances. It appears to be a good illustration of the positive collaboration of two types of improvements which, a priori, seemed contradictory.

REFERENCES


