Branch-and-Price Approach to the Vehicle Routing Problem with Time Windows

Lloyd A. Fasting

May 2014

Supervisors:

dr. M. Firat
dr.ir. M.A.A. Boon
J. van Twist MSc.
Contents

1 Introduction .................................................. 1
   1.1 Problem description .................................. 1
   1.2 Solution method ...................................... 2

2 Software implementation .................................. 5
   2.1 Description of the implementation ................. 5
      2.1.1 Initializing the routes ......................... 5
      2.1.2 Restricted Master Problem ..................... 6
      2.1.3 Pricing Problem: Finding a minimum-cost route in the modified network 7
      2.1.4 Branch-and-Bound search ....................... 7
      2.1.5 Performance guarantee: Relative error ........ 9

3 Results .................................................. 10
   3.1 Performance of Column Generation ................. 10
   3.2 Computation times of Branch-and-Price ........... 11
   3.3 Conclusion ........................................... 12
Chapter 1

Introduction

This report is the result of a three month internship at Quintiq, 's-Hertogenbosch. The internship is part of graduation project. The value of this report lies first and foremost in the foundations it lays for the continuation of the research.

The structure of this report is as follows. First a description of the problem is given, followed by the solution method that is used for solving the problem. The remaining part of the report gives a description of the software implementation of this method.

1.1 Problem description

There is a fleet of identical vehicles with a fixed capacity located at a depot, and a set of customers $N$. Every customer $i \in N$ has to be visited exactly once within a predefined time window $[l_i, u_i]$ by one of the vehicles. $l_i$ and $u_i$ are called the release date and due date of customer $i$, respectively. The service time at a customers is denoted with $s_i$. This report deals with hard time windows. When a vehicle arrives at a customer before its release date it is required to wait until the release date before starting service. A vehicle is not allowed to arrive after the due date.

The problem has the following two objectives, stated in decreasing order of importance: minimizing the number of vehicles, and minimizing the total travel distance. Each vehicle starts and ends at the depot. We search for a set of feasible vehicle routes, such that the two objectives are minimized in order of importance. A route is characterized by an ordered list of customers that are visited in the given order. This problem is known as the Vehicle Routing Problem with Time Windows (VRPTW).

An example of an instance is shown in figure 1.1. At every customer the time window is shown between “[“ and “]” and the demand in bold face. The vehicles have a capacity of 50. What is the minimum number of vehicles required to visit all customers? Since the sum of the demands equals 80, we need at least two vehicles. But more vehicles can be required. For example, it may not be possible to visit some customers with the same vehicle because of their time windows.
Chapter 1. Introduction

Figure 1.1: Example of a VRPTW instance with a vehicle capacity of 50

Figure 1.2: A feasible solution to the VRPTW

Figure 1.2 shows an example of a feasible solution. Since the cardinality of the set of possible solutions is not that large we can manage to find the optimal solution by hand. We can also verify its optimality. However as the number of customers becomes larger, this is not possible anymore. The number of solutions grows extremely fast. Therefore algorithms are needed to solve larger instances of the VRPTW. Such an algorithm will be the main topic of this report.

In fact, the VRPTW is NP-hard. This can be derived as follows: The VRPTW is a generalization of the TSP, which is NP-hard, so also the VRPTW must be NP-hard.

1.2 Solution method

To efficiently solve the problem a reformulation of the problem is needed. A popular and effective formulation uses routes. For example the following articles all use a route formulation: [1], [2], [3], [4]. A route is a sequence of customers visited by one vehicle. A route formulation is preferred because the LP-relaxation of a problem with route variables delivers better lower bounds than one with arc variables. To gain intuition about this statement one could think of routes as objects having more structure than arcs. In the proceedings of this report we will see that relaxing the problem is used for solving it. Relaxing arcs would deliver solutions containing an unstructured collection of partially selected arcs, whereas routes maintain a sense of ordering even when selected partially.

A disadvantage of using a route formulation is the following: The set of all routes, $R$, is extremely large. Therefore we only consider a subset $R' \subset R$, and generate new routes when necessary. This approach is known as Column Generation. The structure of Column Generation consists of a Master Problem and a Sub Problem. The master problem is the actual problem we want
to solve to optimality. Its mathematical formulation is shown below.

\[
\min \sum_{r \in R} c_r y_r \quad s.t. \\
\sum_{r \in R} \delta^i_r y_r = 1 \quad i \in N \setminus \{0\} \quad (1.2) \\
\sum_{r \in R} y_r \leq |V| \quad (1.3) \\
y_r \in \{0, 1\} \quad r \in R \quad (1.4)
\]

In this formulation \(c_r\) is the cost of route \(r\), and \(y\) is its selection variable. \(\delta^i_r y_r\) equals 1 if customer \(i\) is served by route \(r\), and 0 otherwise. But it is unsolvable in general because of the large set of routes \(R\). Therefore we define a Restricted Master Problem (RMP) which uses a subset \(R' \subset R\) and has relaxed route variables. Its mathematical formulation is:

\[
\min \sum_{r \in R'} c_r y_r \quad s.t. \\
\sum_{r \in R'} \delta^i_r y_r = 1 \quad i \in N \setminus \{0\} \quad (1.6) \\
\sum_{r \in R'} y_r \leq |V| \quad (1.7) \\
0 \leq y_r \quad r \in R' \quad (1.8)
\]

Constraints (1.6) makes sure every customer is visited exactly once. Constraint (1.7) bounds the number of vehicles used by the size of the fleet \(V\). Constraint (1.8) is written without an upper bound (\(\leq 1\)) since (1.6) already takes care of this.

\[\text{Figure 1.3: Visualization of the Column Generation procedure}\]

![Visualization of the Column Generation procedure](image-url)
The optimal solution will probably contain routes that are not in $R'$. Therefore a Sub Problem is needed for generating new routes that are not yet in set $R'$. New routes are not generated randomly. The RMP provides us with information about which customers are preferred to be used in new route. This information comes in the form of dual values of constraints 1.6. Customers with higher dual have a higher priority of being included in a new route. This is translated to the Sub Problem by modifying the arc costs in the network and finding a feasible route with minimal cost. The arcs costs are modified as follows:

\[ c'_{ij} = c_{ij} - \pi_j, \]

where $c'_{ij}$ is the modified arc cost, $c_{ij}$ the arc length and $\pi_j$ the dual value of customer $j$. The RMP and the Sub Problem are alternated in the Column Generation procedure. A schematic overview can be found in figure 1.3.

When the Sub Problem can not find a route with negative cost, the RMP is solved to optimality. Note that since the RMP is a relaxed version of the main problem the solution will not be integral in general. Therefore we perform a Branch-and-Bound search. Column Generation is used to find a lower bound at every node. This combination of Branch-and-Bound and Column Generation is called Branch-and-Price.
Chapter 2

Software implementation

In the previous chapter we have described the problem and explained the method for solving it. This chapter focusses on the software implementation. The programming language we use is Java. First an overview is given of the structure of the code. Subsequently the different parts are described in more detail.

2.1 Description of the implementation

The main components of the software are the RMP solver, the Sub Problem solver and the Branch-and-Bound search. Furthermore there are components for generating initial routes, finding upper bounds and making the next branching decision. A visualization of the interplay of the components can be found in figure 2.1.

![Figure 2.1: Interplay of the main software components](image)

2.1.1 Initializing the routes

The algorithm can start with an empty set $R'$. In that case every route must be generated by the Sub Problem. To increase the speed of the algorithm we should initialize $R'$ with a set of
initial routes.
There is a balance between the quality of routes in $R'$ and the computation time of determining $R'$. We choose a Dynamical Programming approach. It delivers promising routes that satisfy the capacity and temporal constraints, but requires a considerable amount of running time, especially for instances with a large number of customers. In the end, the time for calculating $R'$ will be well spent because less routes have to be generated during Branch-and-Price.

2.1.2 Restricted Master Problem

IBM ILOG CPLEX is used for solving the Restricted Master Problem (RMP). The RMP solution is a selection of the routes from $R'$ having minimum cumulative cost such that every customer is visited exactly once. The cost of a route is simply the sum of its arc distances. Such a problem is also called a Set Covering. An example of an output can be found in table 2.1.

<table>
<thead>
<tr>
<th>Solution value</th>
<th>Route</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0 2 5 7 6 8 3 1 4 0</td>
</tr>
<tr>
<td>1.0</td>
<td>0 19 18 20 0</td>
</tr>
<tr>
<td>0.333</td>
<td>0 14 15 11 12 9 10 13 17 0</td>
</tr>
<tr>
<td>0.333</td>
<td>0 14 12 15 16 9 10 0</td>
</tr>
<tr>
<td>0.333</td>
<td>0 11 15 16 17 13 0</td>
</tr>
<tr>
<td>0.333</td>
<td>0 14 12 11 16 9 10 13 17 0</td>
</tr>
</tbody>
</table>

Table 2.1: Solution example of the RMP

After a solution is found CPLEX can immediately return the values of the dual variables. As mentioned earlier, the dual value of a customer constraint is a measure of the importance of including that customer in the new route.

Artificial variable

Even with an initial set of routes it may not be possible to cover all customers. In this case the RMP will be infeasible. In order to maintain the feasibility of the RMP, we use an artificial variable with an high cost. Unless the RMP is infeasible, so is the problem instance, the artificial variable cannot be in the basis of any feasible solution due to its high cost.

Minimizing the number of vehicles first

Our problem has two hierarchical objectives; the first one is minimizing the number of used vehicles, and the second one is minimizing the total traveled distance. This can be accomplished by adding a sufficiently large cost $M_0$ to the outgoing arcs from the depot. This cost can be
seen as a setup cost. For $M^0$ we need an upper bound on the length of a route.

$$M^0 = \frac{Q}{q_{\text{min}}} c_{\text{max}},$$

where $q_{\text{min}}$ is the minimum demand, and $c_{\text{max}}$ is the maximum arc distance.

### 2.1.3 Pricing Problem: Finding a minimum-cost route in the modified network

In order to find the routes that will be added to the RMP we solve a Pricing Problem. In the CG framework this is also called the Sub Problem. The target of the Sub Problem is to find feasible routes with negative cost in a network with modified arc costs. This modification uses the dual values coming from the RMP and is described in the previous chapter.

An example of a modified network can be found in figure 2.2. The method we use for finding routes is based on an insertion heuristic. Every route that has negative cost can improve the solution. But when the heuristic method does not find a route anymore, we still need to solve it optimally to be sure no route with negative reduced cost can be found. Assuming that all routes in network 2.2 are feasible there are two routes having negative cost (0-2-1-0 and 0-2-0) that could be added to the RMP.

![Network with modified arc costs](image)

**Figure 2.2:** Network with modified arc costs

### 2.1.4 Branch-and-Bound search

The solution obtained from the Column generation procedure is not integral in general. That is because we solve a relaxed version of the Master Problem. But recall that an optimal solution must have integer route variables, since a vehicle either departs from depot or does not. To obtain an integral solution we perform a Branch-and-Bound search on the solution.
A Branch-and-Bound search can be characterized by a Branch-and-Bound tree. Every node in the tree can have child nodes. Creating those child nodes is called branching. In our case there are two nodes created while branching. This splits the solution space into two disjoint subspaces. The next part will describe the implemented branching rules.

**Branching on the number of vehicles**

Let $\eta$ be the total number of routes used in a solution of the RMP. The number of routes, $\eta^*$, is generally not integral. In that case we decide to branch on the number of vehicles. In one branch we have $\eta \leq \lfloor \eta^* \rfloor$, in the other $\eta \geq \lceil \eta^* \rceil$. This divides the solution space into two disjoint subspaces. Each time a solution of the RMP is not integral, we branch on the number of vehicles.

When the solution of the RMP is integral we branch in a different way. The two other branching decisions are described in the next sections.

**Branching on arcs**

Although the problem is formulated in terms of route variables, we branch on arc variables. The following method is used to derive the solution values of the arcs from the solution values of the routes. Let $x_{ij}^*$ be the solution value of arc $(i,j)$ and $y_r$ the solution value of route $r$. Then it holds that:

$$x_{ij}^* = \sum_{r \in R'} y_r^*$$

Since a structured growth of the routes is preferred when choosing the branching arc, only arcs that leave the depot or connect to any already fixed arc are considered. This enables us to include stochastic travel times in the next part of the project, because arcs need to be connected to the depot to compute arrival time distributions. The arc with highest solution value is chosen. When a solution contains multiple arcs with maximal solution value we use tie-breaking rules to choose one of them. For example, arcs that are in less routes are favored over arcs that are in more routes of the solution. Now arcs are fixed such that routes emerge from the depot. These routes are called partial routes. When a route connects to the depot again it is called a fixed route.

**Branching on routes**

In the solution of the RMP routes may come with high solution values. We would probably branch on all arcs of that route sequentially. Computation time can be saved by branching on
all arcs of that route at once. This is called branching on a route. If the solution value of a route equals 1, and after branching on the first arc it stays 1, we choose to branch on the route.

Bounding

At every node of the search tree we use Column Generation to find a lower bound. Once no routes can be found anymore, the RMP provides us with a lower bound for the optimal solution value. Pruning means a node is not explored, so there is no branching at that node. Three cases can be distinguished:

- If the lower bound solution has only integral route variables the node is pruned because of integrality. We compare its value with the current best integral solution, and update it when the found solution is better.

- If the lower bound solution is not integral the node is only pruned if its lower bound value is higher than the current best integer solution. Otherwise the node is added to the queue of nodes that need to be explored.

- If the solution is infeasible (because the artificial variable is not 0), the node is pruned because of infeasibility.

When the queue of nodes becomes empty the algorithm stops. The current best integral solution is the final solution. This solution is the optimal solution to the problem.

2.1.5 Performance guarantee: Relative error

When computing bounds of nodes errors are made. Moreover CPLEX returns “optimal” solutions that are actually within a $10^{-6}$ range of the true optimal solution. This is called the optimality tolerance. Therefore we also choose an $\epsilon$-range in which we claim $\epsilon$-optimality of the found solution. We take $\epsilon = 10^{-6}$.

Let $v$ be the value of a lower bound solution, and $v^*$ the value of the current best integral solution. Then $v$ is pruned if the following holds:

$$v > v^*(1 - \epsilon)$$

In the end, the final solution lies within an $\epsilon$-neighborhood of the exact optimal solution.
Chapter 3

Results

In this chapter we will show the computational results of the Branch-and-Price method for the VRPTW with deterministic travel times. First we will analyze the Column Generation procedure, after that the Branch-and-Price results are given.

*Note that the current state of the algorithm is an intermediate state. The algorithm will be improved and adjusted in the continuation of the graduation project. Therefore the results shown here only provide a snapshot of the current performance of the algorithm.*

The Solomon instances are well known and widely used for studying the VRPTW. They provide us with a variety of different types of problems. Therefore we use them for testing our Branch-and-Price based algorithm. An example of an instance is c101. If only a subset of the customers is considered we denote this as follows: c101_30. This means only the first 30 customers are taken into account.

3.1 Performance of Column Generation

The Column Generation procedure is an important part of Branch-and-Price. We can gain insight into the convergence of CG by plotting the reduced cost of the generated routes. In figure 3.1 we observe a convergence of the reduced cost. It is not necessarily monotonic.

![Convergence of reduced cost c101_20](image)

**Figure 3.1:** Progression of the reduced cost of the generated routes in the root node of c101_20
The computationally demanding part of CG is solving the Sub Problem. The time for solving the RMP is negligible when compared with the Sub Problem. Two methods are used for generating new routes in the Sub Problem: An exact method using CPLEX that finds the minimal reduced cost route, and an insertion based heuristic.

We expect that the exact method delivers better routes which results in a more monotonic convergence. Also we expect that the heuristic based method is much faster. (Note that solving the Sub Problem heuristically does not effect the exactness of the whole algorithm. Every route with negative reduced cost can lower the RMP objective value. And if the heuristic does not find a route with negative reduced we still use CPLEX to search for the minimal reduced cost route)

Figure 3.2 shows the convergence of the reduced cost of instance c107_30 when the Sub Problem is optimally solved. The computation time is 200.4 seconds. Figure 3.3 shows the convergence of the reduced cost of the same instance, but the routes are now generated heuristically. The computation time is 52 seconds. In both cases the instance is solved in the root node, but the computation times differ significantly. Also we see that the convergence using the exact Sub Problem is more focussed, which is what we expected.

$$\text{Figure 3.2: Reduced cost of c107_30 in which the Sub Problem is optimally solved.}$$
$$\text{Computation time: 200.4 sec.}$$

$$\text{Figure 3.3: Reduced cost of c107_30 (heuristic pricing) in which routes are generated heuristically.}$$
$$\text{Computation time: 52 sec.}$$

### 3.2 Computation times of Branch-and-Price

Table 3.1 shows the computation time in seconds for solving the instance to optimality. Computations are made a on computer with an Intel(R) Core(TM) i5 CPU (2.40 GHz) and 8.00 GB RAM. The columns correspond to the number of customers taken from the instance. We have set a time limit of 200 seconds.
Table 3.1: Computation times in seconds for solving the VRPTW with the Branch-and-Price method

<table>
<thead>
<tr>
<th>Instance</th>
<th>Nr. of customers</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>c101</td>
<td>0.2 4 11 9.8 56</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c102</td>
<td>10 108</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c103</td>
<td>8.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c104</td>
<td>26.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c105</td>
<td>1.4 5.5 37.5 122.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c106</td>
<td>0.8 5.2 14 43 103</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c107</td>
<td>0.4 9.1 25.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c108</td>
<td>5 55 53.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c201</td>
<td>1 3.6 41.1 138 158</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r101</td>
<td>1.9 0.9 1.6 3.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.3 Conclusion

In conclusion we have constructed an exact method for solving the VRPTW that delivers optimal solutions to benchmark instances. Problem formulations, branching rules and bounding methods are defined in order to find solutions within less time. At the moment we can solve specific instances up to approximately 40 customers within reasonable times.

As we can see in table 3.1 the computation times are highly dependent on the instance. Every instance has its own difficulty, which means constructing a general algorithm that is efficient for all types of instances, needs fine-tuning and improvements in the continuation of the project.
Bibliography


