A Branch
Cut algorithm for the Multi-Trip Vehicle Routing Problem with Time Windows
Diego Cattaruzza, Paolo Gianessi

To cite this version:
Diego Cattaruzza, Paolo Gianessi. A Branch
Cut algorithm for the Multi-Trip Vehicle Routing Problem with Time Windows. ROADEF2017 - 18ème Conférence de la Société Française de Recherche Opérationnelle et d’Aide à la Décision, Feb 2017, Metz, France. <hal-01651581>

HAL Id: hal-01651581
https://hal.archives-ouvertes.fr/hal-01651581
Submitted on 29 Nov 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
A Branch&Cut algorithm for the Multi-Trip Vehicle Routing Problem with Time Windows

Diego Cattaruzza\textsuperscript{1}, Paolo Gianessi\textsuperscript{2}

\textsuperscript{1} Université Lille, CNRS, Centrale Lille, Inria, UMR 9189 - CRIStAL
Centre de Recherche en Informatique Signal et Automatique de Lille
F-59000 Lille, France
diego.cattaruzza@ec-lille.fr

\textsuperscript{2} Mines Saint-Étienne, FAYOL-EMSE, LIMOS, UMR CNRS 6158, France
paolo.gianessi@emse.fr

\textbf{Keywords}: Multi-Trip VRP, Time Windows, Branch&Cut, Exact Methods

1 Introduction

The Vehicle Routing Problem with Time Windows (VRPTW) deals with the determination of the trips of a homogeneous fleet of capacitated vehicles based at a central depot to deliver a set of customers, while complying delivery time intervals imposed by these latter. The Multi-Trip VRP with Time Windows (MTVRPTW) goes one step further by allowing vehicles to perform not one but a sequence of trips, called a journey, under a maximum overall shift length constraint. Moreover, the time to load a vehicle is service-dependent, i.e. proportional to the total service time of the subsequent trip. Hence, the MTVRPTW consists in finding a set of back-to-depot delivery trips and assigning them to vehicles so that the total travelled distance is minimized and:

\begin{itemize}
  \item each customer is visited exactly once and service starts within the associated time window;
  \item vehicle capacity is respected;
  \item trips assigned to one vehicle do not overlap in time and do not exceed a maximum duration.
\end{itemize}

2 Short literature review

Recently, MTVRPTW has got the attention of scholars mostly due to its application to city logistics (see [4]). However, as far as we are aware of, the literature of exact methods is still scarce. In [6], an algorithm based on the Branch&Price paradigm is proposed which can solve most of the adapted Solomon instances (derived from the benchmark VRPTW instances first proposed in [8]) with 25 customers and 2 vehicles. In [5], the same authors study a variant where the length of a trip is limited. Another Branch&Price algorithm is proposed in [1], where another variant of the problem is tackled where the visit of customers is not mandatory but rewarded with a profit. Finally, [7] proposes an exact algorithm for a variant with both profits and limited trip duration.

3 Solution method

We propose a two-index MILP formulation for the MTVRPTW that makes use of base and replenishment arcs. The former model the direct connection between two nodes, whereas the latter imply a reload operation between two client nodes. Replenishment arcs, which have been used e.g. in [3], allow to represent a journey as an elementary path with both endpoints in the
do not hallucinate.

<table>
<thead>
<tr>
<th>instance</th>
<th>%r</th>
<th>T_r</th>
<th>T_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>C201_2_25</td>
<td>1.41</td>
<td>1.8</td>
<td>7.5</td>
</tr>
<tr>
<td>C202_2_25</td>
<td>2.30</td>
<td>6.5</td>
<td>616.8</td>
</tr>
<tr>
<td>C203_2_25</td>
<td>0.91</td>
<td>7.3</td>
<td>930.1</td>
</tr>
<tr>
<td>C204_2_25</td>
<td>0.22</td>
<td>9.3</td>
<td>100.6</td>
</tr>
<tr>
<td>C205_2_25</td>
<td>4.35</td>
<td>2.7</td>
<td>71.3</td>
</tr>
<tr>
<td>C206_2_25</td>
<td>2.22</td>
<td>4.1</td>
<td>221.3</td>
</tr>
<tr>
<td>C207_2_25</td>
<td>0.08</td>
<td>4.5</td>
<td>3.2</td>
</tr>
<tr>
<td>C208_2_25</td>
<td>0.73</td>
<td>3.9</td>
<td>379.9</td>
</tr>
<tr>
<td>average</td>
<td>1.53</td>
<td>5.0</td>
<td>291.3</td>
</tr>
</tbody>
</table>

TAB. 1 – Results on clustered instances with 25 customers and 2 vehicles.

4 Computational results

Tests have been conducted on a set of 54 adapted Solomon instances with 25 to 50 customers and a fleet of 2 to 4 vehicles. The algorithm is coded in C++ and makes use of CPLEX 12.6.1. Experiments are run on a Intel Core i7-5500U 2.4 Ghz machine with 15.56Gb RAM. We allocate 30s to solve the root node and extra 1200s after branching on it.

Table 1 shows the results on the class C (clustered) instances of of [4] with 25 customers. For these instances, the optimal value (which is known from [6]) is always found. $T_r$ and $\%_r$ report the root computation time and the gap of the root lower bound with respect to the optimal value, while $T_i$ is the additional running time to explore the Branch&Bound tree.

References