Branch-and-price algorithms for the Bi-Objective Vehicle Routing Problem with Time Windows

Estèle Glize, Nicolas Jozefowiez, Sandra Ulrich Ngueveu

MOPGP 2017
Branch-and-price algorithms for the Bi-Objective Vehicle Routing Problem with Time Windows

Estèle Glize [1,2], Nicolas Jozefowiez [1,2], Sandra Ulrich Ngueveu [1,3]

[1] CNRS, LAAS, [2] INSA Toulouse, [3] INP Toulouse, F-31400 Toulouse, France
{glize,njozefow,ngueveu}@laas.fr

Keywords: *Combinatorial MOP, vehicle routing problem, exact method, column generation*

This paper presents generic exact methods for the bi-objective vehicle routing problems (BOVRP). The first part explains the BOVRP. Then, an efficient mono-objective solution method used as a component of bi-objective methods is introduced. After that, the methods are presented. The computational evaluation is performed on a specific variant of the VRP, the VRP with time windows (VRPTW).

1 Context

The Vehicle Routing Problem was first introduced in 1959 by Dantzig and Ramser [3] which aimed to minimize the cost of gasoline transportation between a bulk terminal and several service stations.

The bi-objective VRP with time windows (BOVRPTW) is a variant of the VRP. Let $G = (V, E)$ be a non-oriented graph. A node $i \in V \backslash v_0$ is called a *customer* and has a demand q_i. These demands are satisfied by a fleet of K vehicles of capacity Q. A vehicle k starts and returns at a node v_0 called the *depot* and performs a route r_k by passing through a set of customers. The route r_k is said to be feasible if each customer i served by the route is visited between the due time dt_i and the end time et_i and if the total capacity of a vehicle is not exceeded.

An edge of the graph has two costs c^1 and c^2. Each route r_k provides two costs c^1_k and c^2_k representing the sum of the two costs on the used edges. The aim of the BOVRP is to minimize the sum of each cost of the routes used.

Let Ω be the set of feasible routes r_k, $k \in [1, .., K]$ and a_{ik} be equal to 1 if the customer i belongs to the route r_k. The Set Partitioning formulation of the BOVRPRTW [2] can be stated as follows:

$$\begin{cases} minimize \ (\sum\limits_{r_k \in \Omega} c^1_k \theta_k, \sum\limits_{r_k \in \Omega} c^2_k \theta_k) \\ \sum\limits_{r_k \in \Omega} a_{ik} \theta_k \geq 1 \quad (v_i \in V \backslash \{v_0\}), \\ \sum\limits_{r_k \in \Omega} \theta_k \leq K, \\ \theta_k \in \{0, 1\} \quad (r_k \in \Omega). \end{cases} \tag{1}$$

where θ_k is a variable that indicates if the route r_k, $k \in [1, .., K]$, is selected in the solution ($\theta_k = 1$) or not ($\theta_k = 0$).

This formulation has an exponential number of variables and a polynomial number of constraints. It is therefore suitable for a column-generation-based solution method such as the method of Baldacci et al. [1] presented in section 2.

The ϵ-constraint method, introduced in section 3, is one of the most efficient exact method for bi-objective integer problems as for the bi-objective prize-collecting steiner tree problem[4]. To compare our methods, the reference method (3.1) reproduced a classical ϵ-constraint.

2 Solution in mono-objective vehicle routing problems

Solving the BOVRP requires a method that optimally solve the mono-objective VRP (MOVRP). One of the state-of-the-art methods is the one of Baldacci et al. [1]: an exact method to solve the MOVRP, based on the set partitioning formulation (1) with only one cost to minimize. This method allows to restrict the number of routes to obtain the optimal integer solution with a three steps algorithm:

1. Compute a good lower bound LB with column generation and a good upper bound UB via a heuristic. Compute the gap $\gamma = UB - LB$.
2. Generate all routes with a reduced cost less or equal to γ. Indeed, it can be proven that routes with a reduced cost higher than γ can not be in the optimal integer solution [4].
3. Solve the initial integer problem on the restricted set of routes from step 2 to obtain the optimal (integer) solution.

As previously mentioned, this method will be used to solve the BOVRP. In the following, we will refer to the step 2 by $\mathcal{BALD}(UB,LB)$ with UB and LB the upper bound and the lower bound previously computed. It returns a restricted set of routes $\overline{\Omega}$.

3 Methods

This section presents three methods that compute the minimum complete Pareto front of the BOVRPTW introduced in section 1. So, all supported and non-supported non-dominated points are given, but only one efficient solution is provided for each one. They are based on the Baldacci et al. method, presented in section 2.

3.1 Reference method

The first method is called the reference method as it is the more direct way to use the Baldacci et al. method in a BOVRP with an ϵ-constraint method.

The algorithm, summarized in Algorithm 1, is based on the ϵ-constraint formulation that aims to minimize the first cost c^1 under the constraint that the second cost has to be lower than a certain value ϵ. This formulation CMP(ϵ) can be stated as follows:

$$\begin{cases} minimize \sum\limits_{r_k \in \Omega} c_k^1 \theta_k \\ \sum\limits_{r_k \in \Omega} c_k^2 \theta_k \leq \epsilon \\ \sum\limits_{r_k \in \Omega} a_{ik} \theta_k \geq 1 \quad (v_i \in V \backslash \{v_0\}), \\ \sum\limits_{r_k \in \Omega} \theta_k \leq K, \\ \theta_k \in \mathbb{N} \quad (r_k \in \Omega). \end{cases} \tag{2}$$

where all notations are the same as in the formulation (1).

The reference method consists in finding LB, the optimal value of the linear relaxation of the problem (2) (LCMP) and UB, a feasible solution of the integer problem (2) (RCMP). Then, it applies $\mathcal{BALD}(UB,LB)$ to obtain $\overline{\Omega}$, the restricted set of routes with their reduced cost within the gap γ. And finally, the algorithm optimally solves the ϵ-constraint formulation (2).

This process is repeated from $\epsilon = +\infty$ to a certain value for which there is no more solution.

Algorithm 1 Algorithm of the reference method

$\epsilon \leftarrow +\infty$
while $\exists$ a solution **do**
  Solve LCMP(ϵ) to obtain LB
  Solve $RCMP(\epsilon)$ to obtain UB
  $\overline{\Omega} \leftarrow \mathcal{BALD}(UB, LB)$
  Solve $CMP(\epsilon)$ on $\overline{\Omega}$ to obtain S_{OPT}
  Set $\epsilon \leftarrow S^2_{OPT} - 1$
end while

3.2 *Generate-one-time*

The second method requires as input:

- LB: the cost c^1 of the optimal solution of the linear relaxation of the problem which minimizes the cost c^1.
- UB: the cost c^1 of the optimal solution of the linear relaxation of the problem which minimizes the cost c^2.

The method applies $\mathcal{BALD}(UB,LB)$ and performs an ϵ-constraint method of the formulation (2) on this new set of routes. It is summarized in the Algorithm 2.

Algorithm 2 Algorithm of *generate-one-time*

$\overline{\Omega} \leftarrow \mathcal{BALD}(UB, LB)$
$\epsilon \leftarrow +\infty$
while $\exists$ a solution **do**
  Solve $CMP(\epsilon)$ on $\overline{\Omega}$ to obtain S_{OPT}
  Set $\epsilon \leftarrow S^2_{OPT} - 1$
end while

3.3 *Two-steps*

This method is based on the parametrized formulation that optimizes a weighted-sum of the two objectives. The formulation is noted PMP(λ):

$$\begin{cases} minimize\ \lambda \sum\limits_{r_k \in \Omega} c^1_k \theta_k + (1-\lambda) \sum\limits_{r_k \in \Omega} c^2_k \theta_k \\ \sum\limits_{r_k \in \Omega} a_{ik} \theta_k \geq 1 \quad (v_i \in V \backslash \{v_0\}), \\ \sum\limits_{r_k \in \Omega} \theta_k \leq K, \\ \theta_k \in \mathbb{N} \quad (r_k \in \Omega). \end{cases} \tag{3}$$

We also need to introduce the call of the second step of Baldacci et al. method for the direction λ : $\mathcal{BALD}(UB,LB,\lambda)$, where the cost of a solution is computed for the weight λ.

It requires as input the list of N points LB (the front of the linear relaxation of the BOVRPTW obtained by the parametric simplex) and their associated dual solutions. It also needs a list of $N+1$ directions λ in which the points LB_i and LB_{i+1}, $i \in [1, .., N-1]$ are optimal.

The method is composed of two steps. First, a non-complete Pareto front of S points is computed and then, the area in which non dominated point could be are explored. The first step iteratively takes a point LB_i, $i \in [1, .., N]$, its dual solutions and its direction λ_i. In this direction, a feasible solution UB_i is computed and $\mathcal{BALD}(UB_i,LB_i,\lambda_i)$ is applied to obtain a non-dominated point. The second step searches all non-dominated points between two successive points found in the first step. The search between two points S_i and S_{i+1}, $i \in [1, .., S-1]$, can be summarized as follows:

Algorithm 3 Second step of *two-steps* method

Require: Point S_i and S_{i+1}
  Compute direction λ_i', gradient between S_i and S_{i+1}
  Solve LPMP(λ_i') to obtain LB_i' (in practice LB_i' is in the initial list of N points LB)
  Solve RPMP(λ_i') to obtain UB_i' (in practice UB_i' is the nadir point)
  $\overline{\Omega} \leftarrow \mathcal{BALD}(LB_i', UB_i', \lambda_i')$
  Solve PMP(λ_i') on $\overline{\Omega}$ to obtain S_i'
  if PMP(λ_i') is feasible **then**
    Repeat algorithm for S_i and S_i' and for S_i' and S_{i+1}
  end if

4 Results

Each method returns the minimum complete set of non-dominated points. To compare the three methods introduced in Section 3, pairs of Solomon's instances have been merged to create BOVRPTW. The implementation is in C++ and the linear problems and the integer problems are solved with Gurobi 7.1. The following table shows the results of the execution with the CPU time in seconds and on a graph with 25 customers. The column called *BALD* represents the number of calls to procedure $\mathcal{BALD}$ and the one called *col* is the average number of generated columns per call.

	Ref.			*generate-one-time*			*two-steps*		
Instance	time	BALD	col.	time	BALD	col.	time	BALD	col.
R101_C101	10	24	735	**2**	1	1192	5	30	420
R101_C201	8	22	745	**2**	1	1224	3	25	438
R105_RC105	43	37	3736	**22**	1	7119	119	66	2505
R105_C201	47	35	3911	**30**	1	6225	108	57	2471
R109_C101	750	26	33149	**318**	1	62 072	2611	56	1492
R109_RC105	1112	48	30559	**259**	1	53 513	4828	82	7666
RC101_C101	15	10	5128	**4**	1	4850	15	17	3423
RC105_R101	248	18	26797	**137**	1	42 602	189	32	8298
RC106_C101	373	14	45113	**35**	1	18 280	243	18	11 985

It demonstrates that the *generate-one-time* is more efficient than the two others and also that *two-steps* method is a lot less efficient. Regarding the number of calls to the procedure $\mathcal{BALD}$, we can notice that *two-steps* method has a very high number of calls, but the number of generated routes is lower than for *generate-one-time* method.

5 Conclusion and perspectives

The results in Section 4 show that *two-steps* method is less efficient, but the resulting integer problems are easier to solve than for the *generate-one-time* method. For larger graphs, the *generate-one-time* method could produce a prohibitively high number of routes. Therefore, the *two-steps* can not be disregarded based only on the results on small graphs, although it remains to be improved, for example using parallelism or a hybridization of the two methods.

References

[1] Roberto Baldacci, Nicos Christofides, and Aristide Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008.

[2] Michel L Balinski and Richard E Quandt. On an integer program for a delivery problem. *Operations Research*, 12(2):300–304, 1964.

[3] George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

[4] Markus Leitner, Ivana Ljubic, and Markus Sinnl. The bi-objective prize-collecting steiner tree problem. Technical report, Technical report, University of Vienna, Austria, 2013.