On the exact solution of the multi-depot open vehicle routing problem

Vinícius Carvalho Soares, Marcos Roboredo

1. Introduction

In this supplementary material, we show stater-of-art branch-and-cut-and-price (BCP) elements are incorporated to the VRPSolver models proposed by us. Here, we only present the main BCP elements used in the VRPSolver models proposed in the paper. For more details about all elements, we refer to Pessoa et al. (2020). We highlight that the notation used in this material was defined in the paper.

The BCP elements discussed in this material are: Rounded Capacity Cuts (RCCs) separator, ng-paths and enumeration. The incorporation of all of them depends on packing sets concept.

1.1. Packing sets

When the user defines the so-called packing sets for a VRPSolver model, then the main state-of-art BCP algorithms components can be activated such as ng-path relaxation, rounded capacity cuts separators, limited-memory rank-1 cutting planes, and elementary route enumeration.

The packing sets can be defined on vertices or arcs. In this paper we present the definition for vertices. Let $\mathcal{P} \subset 2^{V'}$ be a collection of mutually disjoint subsets of V', where $V' = \bigcup_{k \in K} V^k \setminus \{v^k_{source}, v^k_{sink}\}$. We say that the subsets of $\mathcal{P}$ are packing sets if there is at least one optimal solution for the master problem satisfying the constraints (1), where each h^p_v computes the number of times that a vertex v is visited in a path p.

$$\sum_{p \in P} \left(\sum_{v \in S} h^p_v \right) \lambda_p \leq 1, \quad \forall S \in \mathcal{P} \tag{1}$$

According to constraints (1), for a given packing set $S \in \mathcal{P}$, at most one vertex of S appears in the optimal solution and at most one time. The definition of proper packing sets is an important modelling task of the framework user. In VRPs, commonly the routes are modeled as paths on the graphs and the customers are modeled as packing sets.

1.2. Rounded Capacity Cuts separator

The use of packing sets on vertices allows the framework user define Rounded Capacity Cuts (RCCs) separator. To add a RCC separator, the modeler has to define a capacity $\mathcal{Q}$ and a demand function $d : \mathcal{P} \cup \emptyset \rightarrow \mathbb{R}_+$ such that $d(\emptyset) = 0$ and such taht there is an optimal solution (x^*, y^*, λ^*) of the VRPSolver Formulation proposed in the paper such the following conditions are satisfied:

1. $\sum_{v \in p} d(S(v)) \leq \mathcal{Q}, \forall p \in P$, where $S(v)$ is the packing set that contains v ($S(v) = \emptyset$ if v does not belong to any packing set).
2. for all $S \in \mathcal{P}$ such that $d(S) > 0$, the corresponding constraints in (1) should be satisfied with equality by (x^*, y^*, λ^*).

Based on the function d and capacity $\mathcal{Q}$ provided by the user, the following valid inequality represents a Rounded Capacity Cut for a given $\mathcal{S} \subseteq \mathcal{P}$:

$$\sum_{p \in P} h^p_{\mathcal{S}} \lambda_p \geq \left\lceil \frac{\sum_{S \in \mathcal{S}} d(S)}{\mathcal{Q}} \right\rceil, \tag{2}$$

where $h^p_{\mathcal{S}}$ is the number of times that an arc in path $p \in P$ enters in $\mathcal{S}$. The definition of proper and valid RCC separators is a modelling user task. For classical capacity constraint that is considered by the MDOVRP and MDOVRPTW, a RCC separator can be defined by associating the function d with the demand of the customers and by considering $\mathcal{Q}$ equal to the vehicle capactity. This kind of RCC separator is defined for several VRPSolver models in Pessoa et al. (2020).

1.3. ng-paths

In the context of modeling classical Vehicle Routing Problems (VRPs), a notable limitation of linear relaxation is frequently observed in the presence of non-elementary paths in P (set of resource-contrainded paths) that cannot be included in any integer solution. In such instances, it becomes desirable to remove these paths from the definition of P. However, doing so would considerably complicate the pricing subproblems, to the extent of rendering them intractable in numerous cases. A favorable balance between formulation effectiveness and pricing complexity can be achieved by utilizing ng-paths, a concept introduced by Baldacci and Mingozzi (2009).

For each arc $v \in V$ (Set of vertices in the path generator graph), the user define $Ng(v)$ reprsenting the ng set of v. A path may use two vertices belonging to the same elementarity set S, but only if the subpath between those two vertices passes by a vertex v such that $S \notin Ng(v)$.

1.4. Path enumeration

The path enumeration technique involves the systematic enumeration of all potential paths within a specific set P, which have the potential to contribute to an improved solution. By successfully enumerating these paths and incorporating them into a pool, the corresponding pricing subproblem can be efficiently solved through inspection, resulting in time savings. Additionally, conventional path removal techniques utilizing reduced costs can be applied to eliminate paths from the pools. Once the enumeration process has been successfully completed and the total number of paths in the tables has been significantly reduced to a manageable quantity, the formulation can be constrained solely to these paths. This restricted formulation can then be directly solved using a general Mixed-Integer Programming (MIP) solver (In the paper, we use the solver CPLEX). When the enumeration is turned on, the VRPSolver framework try to enumerate all paths such that $\bar{c}(p) < UB$ - LB, where $\bar{c}(p)$ is the reduced cost of the path p, UB is the best known integer solution cost, and LB is the value of the current linear relaxation. The efficiency of the enumeration increases When the user provides good initial upper bounds.

References

Baldacci, R., Mingozzi, A., 2009. A unified exact method for solving different classes of vehicle routing problems. Mathematical Programming 120, 347–380.

Pessoa, A., Sadykov, R., Uchoa, E., Vanderbeck, F., 2020. A generic exact solver for vehicle routing and related problems. Mathematical Programming 183 (1), 483–523.