Plan

Basic experiments:

- Run the D−L model for the CVRP
- Implement the 2CF−CVRP or 2CF−CVRP m. model for the CVRP
- Implement the RCI−Sep separation routine than identifies violated RCI by solution x_{ij}^*
- Strengthen the D−L or 2CF−CVRP (2CF−CVRP m.) formulations with all violated RC inequalities at the root node, i.e., separate and add violated inequalities to the LP relaxation of the formulation until no violated inequality exists

In the end you should obtain something that is called a cut-and-branch algorithm to the problem using a compact formulation of the problem.

Next step: add cuts at the fractional nodes to a compact CVRP formulation. Note that

- D−L is the model for ACVRP requiring $n(n-1)$ binary variables
- 2CF−CVRP (2CF−CVRP m.) is the model for symmetric CVRP requiring $n(n-1)/2$ binary variables
- RCI−Sep is the separation routine for symmetric RCIs:

$$x(\bar{S}, S) \geq 2\lceil \sum_{i \in S} d_i / Q \rceil$$

Conversion?

Conversion: Symmetric RCI−Sep can be used for ACVRP model as follows: when you have the full set of x_{ij} variables, you can find the capacity of the symmetric cut-set based on the asymmetric model as follows

$$x^*\left(\bar{S},\, S\right) = \sum_{i \in \mathcal{C}} (x^*_{0i} + x^*_{i0})\delta_i + \sum_{i,j \in \mathcal{C},\, i<j} (x^*_{ij} + x^*_{ji})(\delta_i + \delta_j - 2\gamma_{ij}), \tag{1}$$

$$\sum_{i \in \mathcal{C}} \delta_i \geq 2, \tag{2}$$

$$\gamma_{ij} \geq \delta_i + \delta_j - 1, \;\; \gamma_{ij} \geq 0, \qquad i, j \in \mathcal{C},\, i < j, \tag{3}$$

$$\gamma_{ij} \leq \delta_i, \qquad i, j \in \mathcal{C},\, i < j, \tag{4}$$

$$\gamma_{ij} \leq \delta_j, \qquad i, j \in \mathcal{C},\, i < j, \tag{5}$$

$$\delta_i \in \{0,\, 1\}, \qquad i \in \mathcal{C}. \tag{6}$$

Notice the input to a callback:

- model and where
- no other input is allowed
- that implies that reference to any $x[i, j]$ variable from the model is not recognized
- for that reason the following statemetns exist $m.\_x = x$, implying that variables x are "wrapped" in model m, so that in a callback $m.\_x[i, j]$ does refer to $x[i, j]$ variable