Branch-and-Cut Schema for EV Charging Station Placement

1 Problem Formulation

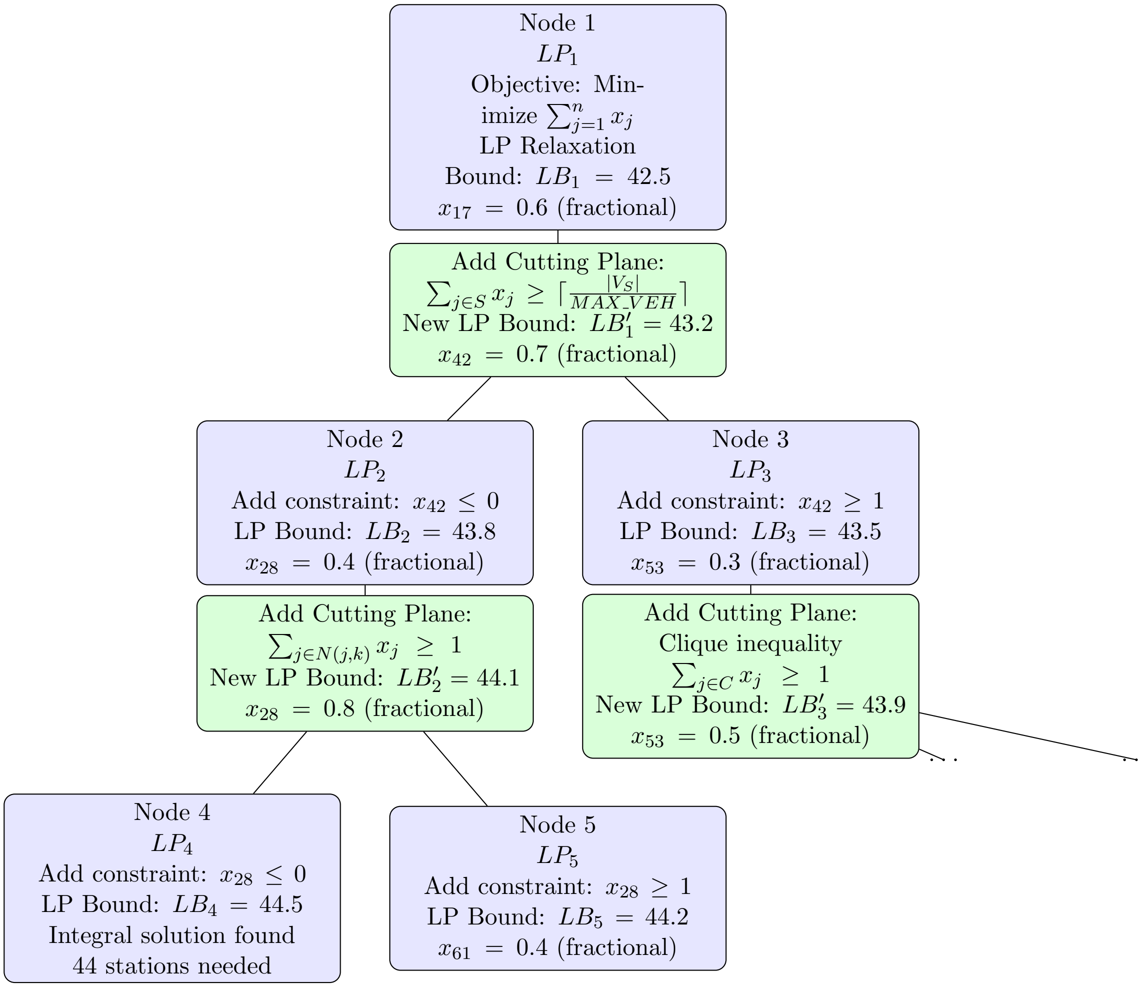
The EV charging station placement problem can be formulated as an Integer Linear Program:

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{j=1}^{n} x_j \\
\text{Subject to:} \quad & \sum_{j=1}^{n} y_{ij} = 1 \quad \forall i \in \{1, \ldots, n\} \\
& y_{ij} \leq x_j \quad \forall i, j \in \{1, \ldots, n\} \\
& y_{ij} = 0 \quad \forall i, j \text{ where } distance(i, j) > \text{BATTERY\_RANGE} \\
& \sum_{i=1}^{n} y_{ij} \leq \text{MAX\_VEHICLES\_PER\_STATION} \cdot x_j \quad \forall j \in \{1, \ldots, n\} \\
& x_j, y_{ij} \in \{0, 1\} \quad \forall i, j \in \{1, \ldots, n\}
\end{aligned}
$$

Where:

- x_j is a binary variable indicating if a charging station is placed at location j
- y_{ij} is a binary variable indicating if vehicle i is assigned to station j

2 Branch-and-Cut Tree



3 Explanation of the Branch-and-Cut Algorithm

Branch-and-cut is an extension of the branch-and-bound algorithm that incorporates cutting planes to tighten LP relaxations. The algorithm works as follows:

3.1 Key Components

1. **LP Relaxation**: Like branch-and-bound, we start by solving the LP relaxation of the integer problem, allowing variables to take fractional values.

2. **Cutting Planes**: Before branching, we try to add valid inequalities (cuts) that:

   - Are satisfied by all integer solutions
   - Cut off the current fractional solution
   - Tighten the LP relaxation, resulting in improved bounds

3. **Branching**: If cuts aren't enough to find an integer solution, we branch by creating subproblems with additional constraints.

3.2 Types of Cuts Used in the EV Charging Station Problem

1. **Covering Cuts**:

$$\sum_{j \in S} x_j \geq \left\lceil \frac{|V_S|}{MAX\_VEH} \right\rceil$$

   Where S is a set of locations and V_S is the set of vehicles that can only be served by stations in S. This ensures we have enough stations to cover the demand.

2. **Neighborhood Cuts**:

$$\sum_{j \in N(j,k)} x_j \geq 1$$

   Where $N(j,k)$ is the set of potential station locations within range of vehicle k. This ensures each vehicle has at least one charging station within its range.

3. **Clique Inequalities**:

$$\sum_{j \in C} x_j \geq 1$$

Where C is a clique of vehicles, meaning none of these vehicles share potential station locations. This ensures we place at least one station for each clique.

3.3 Benefits of Branch-and-Cut over Branch-and-Bound

- **Tighter LP Relaxations**: Cuts strengthen the bounds, making them closer to integer values.
- **Fewer Branches Needed**: With stronger bounds, we can prune more of the search tree.
- **Faster Convergence**: The algorithm typically finds the optimal solution faster than pure branch-and-bound.
- **Better Scalability**: More effective for large-scale problems like our EV charging station placement with thousands of vehicles.

3.4 Example from the Tree

In our example tree:

- At the root node, we add a covering cut that increases the bound from 42.5 to 43.2
- At Node 2, a neighborhood cut increases the bound from 43.8 to 44.1
- At Node 3, a clique inequality improves the bound from 43.5 to 43.9
- Eventually, we find an integer solution requiring 44 charging stations

The cuts helped tighten the relaxations at each node, resulting in a faster convergence to the optimal solution compared to pure branch-and-bound, which found a solution with 45 stations.