Optimization Problem: Mixed BEV and FCEB Fleet for GTFS Routes

1. Objective Function

Minimize the total operational cost:

$$C_{\text{total}} = C_{\text{fleet}} + C_{\text{energy}} + C_{\text{infrastructure}} + C_{\text{maintenance}}$$

Where:

$$C_{\text{fleet}} = \sum_{b \in BEVs} c_{\text{BEV.cost}} + \sum_{b \in FCEBs} c_{\text{FCEB.cost}}$$

$$C_{\text{energy}} = \sum_{t \in T} \left(\sum_{b \in BEVs} x_{\text{charging,b,t}} \cdot c_{\text{electricity}} + \sum_{b \in FCEBs} x_{\text{refueling,b,t}} \cdot c_{\text{hydrogen}} \right)$$

$$C_{\text{infrastructure}} = \sum_{l \in Chargers} x_{\text{opp.chargers,l}} \cdot c_{\text{opp.cost}} + \sum_{l \in RefuelingStations} x_{\text{refueling.stations,l}} \cdot c_{\text{refuel.cost}}$$

$$C_{\text{maintenance}} = \sum_{b \in Buses} c_{\text{maintenance,b}}$$

2. Constraints

2.1 Route Coverage

Ensure every route is served by at least one bus:

$$\sum_{b \in BEVs \cup FCEBs} x_{\text{on.route,r,b,t}} \geq 1, \quad \forall r \in R, \forall t \in T$$

2.2 BEV Energy Balance

Ensure energy balance for BEVs:

$$x_{\text{charging,b,t}} + \sum_{r \in R} x_{\text{on.route,r,b,t}} \cdot c_{\text{energy.use.r}} \leq c_{\text{charge.rate}}, \quad \forall b \in BEVs, \forall t \in T$$

2.3 BEV Range Constraint

Ensure BEVs can complete assigned routes based on battery capacity:

$$c_{\text{route.distance.r}} \leq x_{\text{battery.size,b}}, \quad \forall b \in BEVs, \forall r \in R$$

2.4 FCEB Hydrogen Balance

Ensure hydrogen balance for FCEBs:

$$x_{\text{refueling,b,t}} + \sum_{r \in R} x_{\text{on.route,r,b,t}} \cdot c_{\text{energy.use.r}} \leq c_{\text{refuel.rate}}, \quad \forall b \in FCEBs, \forall t \in T$$

2.5 FCEB Range Constraint

Ensure FCEBs can complete assigned routes based on hydrogen tank capacity:

$$c_{\text{route.distance.r}} \leq x_{\text{hydrogen.tank.size,b}}, \quad \forall b \in FCEBs, \forall r \in R$$

2.6 One Task Per Bus

A bus can perform only one task (route, charging, refueling) at a time:

$$\sum_{r \in R} x_{\text{on.route,r,b,t}} + x_{\text{charging,b,t}} + x_{\text{refueling,b,t}} \leq 1, \quad \forall b \in BEVs \cup FCEBs, \forall t \in T$$

2.7 Depot Charging Capacity

Ensure depot chargers are not overloaded:

$$\sum_{b \in BEVs} x_{\text{charging,b,t}} \leq x_{\text{depot.chargers}}, \quad \forall t \in T$$

2.8 Opportunity Charging Capacity

Ensure opportunity chargers are not overloaded:

$$\sum_{b \in BEVs} x_{\text{charging,b,t}} \leq x_{\text{opp.chargers,l}}, \quad \forall l \in Chargers, \forall t \in T$$

2.9 Refueling Station Capacity

Ensure refueling stations are not overloaded:

$$\sum_{b \in FCEBs} x_{\text{refueling,b,t}} \leq c_{\text{refuel.capacity,l}}, \quad \forall l \in RefuelingStations, \forall t \in T$$

2.10 Fleet Size

Ensure the total fleet size matches the number of buses available:

$$\sum_{b \in BEVs} x_{\text{fleet.BEVs}} + \sum_{b \in FCEBs} x_{\text{fleet.FCEBs}} = \text{Total Fleet Size}$$

3 Time Complexity Analysis

The time complexity of the column generation framework is influenced by three primary components: the Restricted Master Problem (RMP), the Pricing Problem, and the iterative nature of the column generation process.

3.1 Restricted Master Problem (RMP)

The RMP involves solving a linear programming problem over a subset of variables, referred to as columns. The computational complexity of solving a linear program depends on the number of variables (C) and constraints. For the RMP, C represents the number of active columns, which corresponds to feasible bus-route assignments. Standard solvers, such as those based on the simplex or interior-point methods, exhibit a worst-case complexity of:

$$O(C^3)$$

This cubic growth arises due to the linear programming solver's reliance on matrix operations, such as factorization.

3.2 Pricing Problem

The Pricing Problem is responsible for identifying new columns (bus-route assignments) to improve the RMP solution. This involves evaluating all feasible bus-route combinations while adhering to constraints such as timing, energy capacity, and charging/refueling limits. The complexity of this step scales linearly with the number of buses (B) and routes (R), resulting in a computational complexity of:

$$O(B \cdot R)$$

per iteration.

3.3 Column Generation Iterations

Column generation operates iteratively, alternating between solving the RMP and the Pricing Problem until convergence is achieved. The number of iterations required (I) typically depends on the problem's structure and the rate of convergence. Empirical evidence suggests that I often grows logarithmically with the number of routes, i.e.,:

$$O(\log R)$$

This behavior is attributed to the diminishing marginal impact of adding new columns as the solution approaches optimality.

3.4 Overall Time Complexity

The overall complexity of the column generation framework can be expressed as:

$$O\left(I \cdot (C^3 + B \cdot R)\right)$$

where:

- I is the number of iterations.
- C is the number of columns in the RMP, which increases with the number of routes.
- B is the number of buses.
- R is the number of routes.

For large-scale problems, C is proportional to R, and I is approximated as $O(\log R)$. Therefore, the time complexity simplifies to:

$$O\left(\log R \cdot (R^3 + B \cdot R)\right)$$

3.5 Dominant Term

Among the components, the R^3 term from solving the RMP typically dominates for large R, making the overall framework computationally intensive for problems with many routes. While the Pricing Problem contributes linearly to the complexity $(B \cdot R)$, its impact is overshadowed by the cubic growth of the RMP.

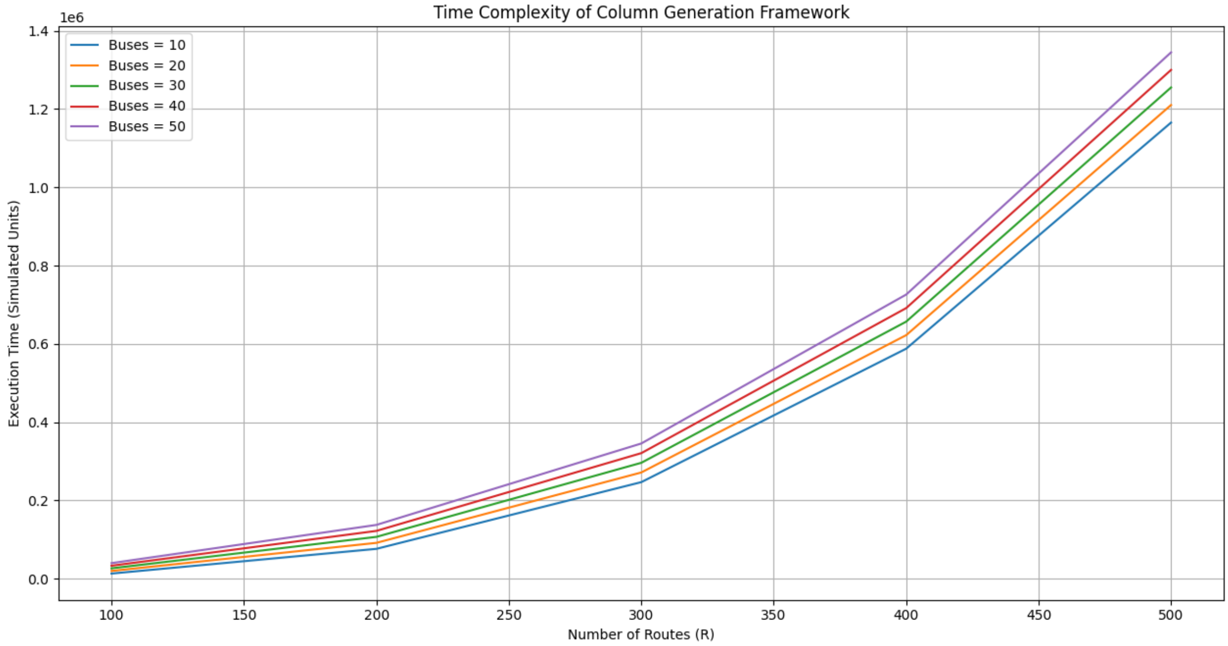

Figure 1: Time Complexity Graph for Column Generation

Complexity Analysis of the Simulated Annealing Algorithm

- **Iterations**:
  - The algorithm runs for max_iterations (let's call it I) or until the temperature drops below a certain threshold.
  - **Complexity**: $O(I)$
- **Neighbor Generation** (`generate_neighbor`):
  - This involves choosing a random route and assigning a new bus.
  - **Complexity**: $O(1)$, as it performs a constant-time operation for each route.
- **Cost Calculation** (`calculate_cost`):
  - This is the most computationally intensive function. It involves:
    * Iterating over all routes to calculate operational costs and check constraints.

    * For each bus, summing up energy consumption across routes for energy balancing.
    * Checking charger and refueling station constraints.
  - Let:
    * R = Number of routes.
    * B = Number of buses.
    * O = Number of opportunity chargers.
    * F = Number of refueling stations.
  - **Complexity**:
    * Route iteration: $O(R)$.
    * Charger utilization: $O(B)$.
    * Energy balancing: $O(B \times R)$ (in the worst case, every bus checks every route).
    * Opportunity charger checks: $O(R \times O)$.
    * Refueling station checks: $O(R \times F)$.
  - **Combined complexity**: $O(R \times (B + O + F))$

- **Acceptance Probability Calculation**:
  - Exponential computation and random number generation are $O(1)$.