Traffic-Aware EV Routing with Charging Stops using ALNS

1 Problem Definition

We solve a single-vehicle Electric Vehicle Routing Problem (EVRP) from an origin $O \in \mathbb{R}^2$ to a destination $D \in \mathbb{R}^2$, possibly via multiple charging stations chosen from a finite set $\mathcal{S} = \{s_1, \ldots, s_n\}$, where each station has coordinates $(\mathrm{lat}, \mathrm{lon})$. Travel times are *traffic-aware* and obtained from the Google Directions API (using `duration_in_traffic` when available).

A route is represented as an ordered sequence of charging stops:

$$R = [O, s_{i_1}, s_{i_2}, \ldots, s_{i_k}, D].$$

2 Traffic-Aware Travel Time Model

For any leg $i \to j$, the Directions API returns:

- distance d_{ij} in meters,
- travel time t_{ij} in seconds (traffic-aware).

We treat each leg as traffic-optimal given the departure time.

3 Energy and SOC Feasibility Model

Let battery capacity be B (kWh) and energy consumption rate be e (Wh/km). Convert distance to kilometers:

$$d_{ij}^{km} = \frac{d_{ij}}{1000}.$$

Energy required for leg $i \to j$:

$$E_{ij} = d_{ij}^{km} \cdot \frac{e}{1000} \quad \text{(kWh)}.$$

SOC drop on the leg:

$$\Delta\mathrm{SOC}_{ij} = \frac{E_{ij}}{B}.$$

With SOC at departure SOC_i and minimum buffer $\mathrm{SOC}_{\min}$, feasibility requires:

$$\mathrm{SOC}_i - \Delta\mathrm{SOC}_{ij} \geq \mathrm{SOC}_{\min}.$$

If violated for any leg, the route is infeasible.

4 Charging Time Model

Assume charger power P_c (kW). If the vehicle arrives with SOC_{arr} and departs with SOC_{dep}:

$$E_c = B(\mathrm{SOC}_{dep} - \mathrm{SOC}_{arr}) \quad \text{(kWh)},$$

$$t_c = \frac{E_c}{P_c} \cdot 3600 \quad \text{(seconds)}.$$

Charging policy. In the baseline implementation, a fixed target SOC is used (e.g., $\mathrm{SOC}_{dep} = \mathrm{SOC}_{target}$). In a more realistic variant, the target is computed from the next leg demand:

$$\mathrm{SOC}_{dep} = \min\left(\mathrm{SOC}_{\max},\ \mathrm{SOC}_{needed\ next} + \mathrm{SOC}_{\min}\right).$$

5 Charging Availability Risk Model

The backend provides a probability of successful availability $p(s,t) \in (0,1]$ for station s at time t. We convert this into a convex risk penalty using negative log-likelihood:

$$\mathrm{RiskPenalty}(s,t) = \lambda \cdot \left(-\ln\left(p(s,t)\right)\right),$$

where $\lambda > 0$ scales the risk into time-equivalent units (seconds).

6 Objective Function

For route R, the objective minimized is:

$$\min_{R} \underbrace{\sum_{(i\to j)\in R} t_{ij}}_{\text{traffic-aware driving time}} + \underbrace{\sum_{s\in R\cap\mathcal{S}} t_c(s)}_{\text{charging time}} + \underbrace{\sum_{s\in R\cap\mathcal{S}} \lambda\left(-\ln(p(s,t_s))\right)}_{\text{risk penalty}}.$$

Subject to SOC feasibility constraints on every leg.

7 Feasible Initialization via Reachability Graph

ALNS requires an initial feasible solution. We build a directed graph

$$V = \{O\} \cup \mathcal{S} \cup \{D\},$$

where an edge $i \to j$ exists if

$$\mathrm{SOC}_{avail}(i) - \Delta\mathrm{SOC}_{ij} \geq \mathrm{SOC}_{\min},$$

with

$$\mathrm{SOC}_{avail}(i) = \begin{cases} \mathrm{SOC}_0, & i = O, \\ \mathrm{SOC}_{target}, & i \in \mathcal{S}. \end{cases}$$

Edge weight is

$$w_{ij} = t_{ij} + \mathbf{1}_{\{j\in\mathcal{S}\}} \cdot \mathrm{RiskPenalty}(j, t_{arr}).$$

We run Dijkstra's algorithm from O to D to obtain a feasible stop sequence.

8 ALNS Metaheuristic

8.1 Solution Representation

A solution is an ordered list of stops:

$$x = [s_{i_1}, \ldots, s_{i_k}],$$

inducing the route $[O, x, D]$.

8.2 Destroy Operator (Where It Is Used)

Given current (or best) stop list x, the **destroy** operator removes r stops uniformly at random:

$$x^{(d)} = \text{Destroy}(x) = x \setminus \{\text{random } r \text{ indices}\},$$

where $r \in [\texttt{DESTROY\_MIN}, \texttt{DESTROY\_MAX}]$.

Exactly where in code. In `alns_optimize()`, destroy is applied here:

```
base = best_stops if random.random() < 0.3 else current_stops
partial = destroy(base) # <-- DESTROY OPERATOR USED HERE
```

8.3 Repair Operator (Where It Is Used)

The **repair** operator attempts to restore feasibility and improve cost by inserting candidate stations:

$$x^{(r)} = \text{Repair}(x^{(d)}),$$

by trying candidate stations (from the backend) and insertion positions, and selecting the best feasible repaired solution:

$$x^{(r)} = \arg \min_{x' \in \mathcal{N}(x^{(d)})} C(x') \quad \text{s.t. } x' \text{ feasible.}$$

Exactly where in code. In `alns_optimize()`, repair is applied immediately after destroy:

```
candidate_stops = repair(partial, stations, departure_time) # <-- REPAIR OPERATOR USED
    HERE
cand = evaluate_solution(candidate_stops, stations, departure_time)
```

8.4 Acceptance Criterion (Simulated Annealing)

Let current solution cost be $C(x)$ and candidate cost be $C(x')$. Accept x' if:

$$C(x') < C(x) \quad \text{or} \quad \exp\left(-\frac{C(x') - C(x)}{T}\right) > U(0,1),$$

where T is the temperature and $U(0,1)$ is uniform random. Temperature is cooled geometrically: $T \leftarrow \alpha T$.

9 Route Evaluation Procedure

Given a stop list x, evaluation simulates:

1. for each leg: query traffic-aware t_{ij}, d_{ij} from Google,
2. compute SOC drop using $\Delta\text{SOC}_{ij} = E_{ij}/B$,
3. reject if SOC violates buffer,
4. if arriving at station: query availability $p(s,t)$ and add risk penalty,
5. compute charging time t_c using charger power.

10 Outputs

The algorithm outputs:

- the chosen stop sequence (charging stations),
- per-leg traffic-aware durations and distances,
- SOC timeline at each route node,
- total objective cost decomposed into driving, charging, and risk.