Due-Job Clustering & Priority Scheduling — Stepwise Formulation

Project Cleo – Field Ops

October 3, 2025

Purpose

Schedule due customers in a data-driven way that produces geographically tight routes while respecting plan cadence, leeway, skills/licensing, customer value, and operational capacity.

Sets & Inputs (over planning horizon D)

- $\mathcal{J}$: due jobs; $\mathcal{Z}$: zones; $\mathcal{A}$: areas; $\mathcal{S}$: service centers.
- For $j \in \mathcal{J}$: location $(\text{lat}_j, \text{lon}_j)$, zone $z(j)$, area $a(j)$, service center $s(j)$.
- Due/leeway: hard due date D_j^{due}; flexible window $[D_j^{\text{min}}, D_j^{\text{max}}] \subseteq D$.
- Cadence by plan: $\text{cad}(j) \in \{30, 60, 90, \dots\}$ days; last service date D_j^{last}.
- Time windows, duration estimate dur_j, required skills/licenses $\mathcal{L}_j$.
- Business attributes: contract value V_j, payment likelihood $p_j^{\text{pay}} \in [0, 1]$, recent cancel/reschedule counts.
- History flags: last service was reservice? $\text{isResvc}_j \in \{0, 1\}$.
- Context: today D_{now}; forecasted shift capacity per zone/day; urbanity class (affects radii).

Derived Quantities

- **Cadence fit at day d:**

$$U_2(j, d) = 1 - \min\left(\frac{\left|(d - D_j^{\text{last}}) - \text{cad}(j)\right|}{\tau_{\text{cad}}}, 1\right)$$

  where τ_{cad} is a tolerance (e.g., $0.2 \times \text{cad}(j)$).

- **Window urgency at day d:**

$$U_1(j,d) = \left(\operatorname{clip}\left(\frac{d - D_j^{\min}}{D_j^{\max} - D_j^{\min}},\ 0,\ 1\right)\right)^{\gamma}, \quad \gamma > 1.$$

- **Business value (normalized):**

$$B(j) = 0.6\ \min\left(\frac{V_j}{V_{90}}, 1\right) + 0.4\, p_j^{\text{pay}}, \quad B_{\text{adj}}(j) = B(j) \cdot (1 - \operatorname{clip}(w_c C_{30} + w_r R_{30},\ 0,\ c_{\max}))$$

  with C_{30}, R_{30} recent cancel/reschedule counts and weights w_c, w_r.

- **Quality boost:** $Q(j) = 1 + \beta_{\text{resvc}} \cdot \text{isResvc}_j$.

Local Density & Neighbor Graph

Let $\mathcal{N}_r(j,d)$ be jobs within radius r km of j whose chosen day is d or whose leeway includes d.

$$N_d(j,d) = \operatorname{clip}\left(\frac{|\mathcal{N}_r(j,d)|}{N_{\text{ref}}},\ 0,\ 1\right)$$

where N_{ref} is a typical route subcluster size (e.g., 8–10).

Composite Priority Score

Weights (w_U, w_B, w_Q, w_N) default to $(0.55, 0.25, 0.10, 0.10)$:

$$U(j,d) = 0.7\, U_1(j,d) + 0.3\, U_2(j,d), \qquad S(j,d) = w_U\, U(j,d) + w_B\, B_{\text{adj}}(j) + w_Q\, Q(j) + w_N\, N_d(j,d).$$

Day Selection with Cluster-Aware Nudging

For each j:

1. Evaluate $S(j,d)$ for all $d \in [D_j^{\min}, D_j^{\max}]$ and set $d^\star = \arg\max_d S(j,d)$.
2. Search nearby days d' with $|d' - d^\star| \le \Delta_{\max}$ (e.g., 1–3 days) and accept d' if

$$N_d(j,d') - N_d(j,d^\star) \ge \tau_N \quad \text{and} \quad S(j,d^\star) - S(j,d') \le \varepsilon_S,$$

   then update $d^\star \leftarrow d'$.

3. If capacity on $d^\star$ is saturated for zone $z(j)$, consider next-best d by descending $S(j,d)$ subject to the same rule.

Geo Bucketing & Cluster Formation (per day and zone)

1. **Pre-bucket:** Partition jobs by geohash at precision yielding ∼0.5–2 km cells (urban) or 2–4 km (rural).
2. **Density merge:** Iteratively merge adjacent buckets if merged radius $\leq r_{\max}$ and size $\leq C_{\max}$.
3. **Sparse handling:** Buckets with size $< C_{\min}$ are parked for the "first-in-area" fallback (below) or are attached to the nearest dense cluster if distance $\leq r_{\text{attach}}$ and cluster constraints remain feasible.

First-in-Area Fallback (anchor selection)

If a job is the first (no dense neighbors) for its area on day d:

$$\text{Anchor}(j) := \begin{cases} \text{zone centroid} & \text{if close enough and zone non-remote,} \\ \text{area centroid} & \text{else if appropriate,} \\ \text{service center} & \text{if both centroids too far,} \\ \text{customer location } (j) & \text{for flagged remote pockets.} \end{cases}$$

Seed a new cluster at the chosen anchor, then *attract* nearby eligible jobs by day nudging (within $\Delta_{\max}$ and ε_S).

Capacity Fit & Routing Handoff

1. For each cluster c, check $\sum_{j \in c} \text{dur}_j$ vs shift capacity for day d and zone z.
2. If oversubscribed, split c by k-means (k=2) respecting time windows and skill/licensing feasibility.
3. Build a routing request (e.g., VROOM) per cluster with:
   - Technician skill/license filters satisfying all $\mathcal{L}_j$,
   - Time windows, depot/start location (team base or cluster centroid for meet-in-field),
   - Anti-waiting settings (small max wait penalties).

Parameters & Defaults

- Radii: r=1–2 km (urban), 3–6 km (rural). Merge cap $C_{\max}$=12–14, min size $C_{\min}$=3.
- Nudging: $\Delta_{\max}$=2 days, τ_N=0.2, ε_S=0.05.
- Weights: (w_U, w_B, w_Q, w_N)=$(0.55, 0.25, 0.10, 0.10)$; $\beta_{\text{resvc}} \in [0.1, 0.25]$.

KPIs & Outputs

For each day and zone:

- Cluster set with centroids, radii, counts, and total duration.
- Priority-ordered job lists by cluster.
- Coverage vs capacity, average cluster radius, added travel vs alternative day, SPH, on-time %, wait minutes.

Pseudocode (high-level)

```
for d in planning_horizon:
  pools = build_day_pools(d, skills/licensing, time windows)
  for j in pools: choose d* via S(j,·), apply cluster-aware nudging
  for z in zones:
    seeds = geohash_buckets(pools[z])
    clusters = density_merge(seeds, r_max, Cmax, Cmin)
    for sparse in parked(seeds, clusters):
      anchor = choose_anchor(z, area, service_center, sparse)
      seed_cluster(sparse, anchor); attract_neighbors_via_nudging()
    clusters = fit_to_capacity(clusters, shifts[z,d])
    routes = route_each_cluster(clusters, constraints)
    persist(routes, clusters, scores)
```

Notes

This formulation prioritizes urgency and cadence while opportunistically snapping jobs to denser days to thicken clusters, producing tighter routes without violating SLAs or business constraints.