Proposed Methodology (ALTAR-H): Attentive, Lexicographic, Temporal-Aware Routing with Hyperclique Priors

We propose **ALTAR-H**, a CPU-only heuristic for Vehicle Routing with Time Windows (VRPTW) tailored to low-memory machines (≤12 GB RAM, no GPU). ALTAR-H integrates: (i) **dual spatial–temporal potentials** to build a high-quality "giant tour" skeleton; (ii) **temporal-feasibility graphs** and **hyperclique priors** that mine short, mutually feasible customer sequences; (iii) **dynamic programming (DP) splitting** and **capacity/time consolidation**; and (iv) a **route-count–first local search** with zero-lateness guards and a route-elimination mechanism. The pipeline is modular and reproducible, mirroring the clarity of recent methodology sections that combine data-driven insights with metaheuristics (we emulate the structured, two-phase exposition style of the attached reference while pursuing a distinct algorithmic path).

1. Problem Setting and Objective

Given a depot (0) and customers ($V=\{1,\dots,n\}$) with coordinates ((x_i,y_i)), time windows ($[a_i,b_i]$), service times (s_i), and demands (q_i), find a set of depot-to-depot routes such that each customer is visited exactly once, vehicle load never exceeds capacity (Q), and all arrivals respect time windows (no lateness). The **lexicographic objective** is:

1. minimize number of routes (vehicles), then
2. minimize total Euclidean distance.

All feasibility checks use **earliest-arrival-with-waiting** updates; lateness is strictly disallowed (hard constraint).

2. Pipeline Overview

ALTAR-H proceeds in five stages:

1. **Dual-Potential Construction & APCA Skeleton**
   Build spatial and temporal potentials, then construct a giant tour via **Alternating Potential Competitive Attention (APCA)**.
2. **Temporal Feasibility Graph (TFG) & Hyperclique Mining**
   Create a directed feasibility graph and mine small **hypercliques** (size 3–5) that are simultaneously feasible sequences; store at least one feasible order per hyperclique.
3. **Tour Shaping & Candidate Generation**
   Generate several time-window-aware giant tours (angle–time sweep, clustering-time order, TW-aware nearest neighbor) and refine each with a light **TW-2opt**.
4. **Split-DP and Consolidation (DSTB)**
   For each tour, run **DP splitting** (lexicographic) into feasible routes; consolidate using depot-sandwich time buffers and clique-respecting merges.
5. **Route-Count–First Local Search (RCF-LS)**
   Apply intra-route 2-opt/Or-opt, inter-route relocate/swap, and an **ejection-reinsertion route-elimination** pass; maintain zero lateness and capacity feasibility.

3. Dual Potentials & APCA Skeleton

3.1 Spatial potential

Compute Euclidean distance matrix (D_{ij}) and normalize:

$$\widehat{D}_{ij} = \frac{D_{ij}}{\max_{u,v} D_{uv}} \in [0,1].$$

3.2 Temporal potential

From depot time (a_0), define the **temporal tension** for arc ($i\!\to\! j$):

$$\tau_{ij} = \max\{0,\, a_j - (t_i + s_i + D_{ij})\} \quad \text{(early wait)},$$

and its normalized complement ($\widehat{\tau}_{ij}=1-\frac{\tau_{ij}}{\max \tau}$), so edges that **leave little slack** get higher temporal pressure.

3.3 Composite attention cost

$$C_{ij} = \lambda\,\widehat{D}_{ij} + (1-\lambda)\,\widehat{\tau}_{ij}, \quad \lambda \in [0,1].$$

A small (λ) emphasizes time feasibility; larger (λ) emphasizes compactness.

3.4 APCA giant tour construction

Starting at the depot, APCA alternates between (A) **temporal-first** attention (choose (j) minimizing (C_{ij}) among feasible (j)) and (B) **spatial-first** attention, with a cooling schedule that gradually shifts weight toward temporal safety as the tour grows. Short tabu lists prevent 2-cycles; capacity is checked cumulatively. The output is a **skeleton order** over all customers.

Rationale: APCA greedily aligns the tour with both proximity and time-window pressure, producing a split-friendly order.

4. Temporal Feasibility Graph & Hypercliques

4.1 TFG construction

Build a directed graph ($G=(V,E)$) where ($(i,j)\in E$) iff serving (i) and driving to (j) allows an **earliest feasible arrival** at (j) with **zero lateness** (under cumulative load and service). We also precompute **KNN lists** in spatial and temporal spaces to bound candidate neighborhoods.

4.2 Hyperclique mining (3–5 nodes)

Enumerate small candidate sets (S) from neighborhood unions and retain those admitting **at least one topological order** ($(v_1,\dots,v_k)$) that is time-feasible (forward simulation) and capacity-safe. Store $(S, \text{one feasible order})$. These **hypercliques act as priors**: short "phrases" likely to remain together during split and search.

Rationale: Hypercliques capture mutually consistent micro-sequences that reduce destructive moves and guide consolidation.

5. Tour Shaping & Candidate Generation

We generate several tour candidates:

- **Angle–Time Sweep (ATS):** bucket by polar angle around the depot; sort within buckets by (a_j,b_j).
- **K-cluster Time (KCT):** farthest-point clustering in (x,y); visit clusters by center angle; order within clusters by time.
- **TW-aware NN (TW-NN):** greedily pick next j with minimal predicted lateness (tie-break by due time then distance).
- **TW-2opt refinement**: a lightweight 2opt pass that minimizes a "TW-conflict score" along the tour.

This set plus the APCA skeleton(s) feed the DP splitter.

6. Split-DP and DSTB Consolidation

6.1 DP splitting (lexicographic)

For a fixed tour $(u_1,\dots,u_n)$, precompute segment tables $F(i,j)$ that indicate whether $u_i\!\to\!\dots\!\to\!u_j$ is feasible and the cost to close to depot. Then solve:

$$
\min \big(\#\text{routes},\, \text{total distance}\big)
$$

via 1-D DP over segment endpoints; keep the split with **fewest routes** (and least distance). We evaluate all tour candidates and select the best split.

6.2 DSTB consolidation

With feasible routes in hand, apply **Depot-Sandwich Time Buffering**: attempt pairwise merges $R_a\cup R_b$ if cumulative demand $\le Q$ and the concatenation stays zero-late after inserting **buffer waits** at the depot boundaries. Respect stored hyperclique orders to avoid breaking known feasible phrases.

Rationale: DP ensures contiguous, feasible blocks; DSTB converts "many short routes" into fewer balanced ones when slack allows.

7. Route-Count–First Local Search (RCF-LS)

We then run a bounded-time local search that **lexicographically** optimizes ((#\text{routes}, \text{distance})) subject to zero lateness:

- **Intra-route**: 2-opt; Or-opt (block sizes 1/2/3).
- **Inter-route**: relocate (1→0), swap (1↔1) with feasibility checks by forward simulation.
- **Route elimination:** choose a small route; **eject** its customers and greedily reinsert them into other routes by best-feasible insertion. If all reinsertions succeed, the route disappears.

We iterate until no improving move is found or a time budget is reached. Hypercliques bias candidate positions during insertion/relocation (try clique-consistent slots first).

8. Feasibility Engine and Guards

All moves use a **single-pass forward simulator**:

$$t_j = \max\{a_j,\, t_i + s_i + D_{ij}\}, \quad \text{lateness} = \max\{0,\, t_i + s_i + D_{ij} - b_j\}.$$

A move is admissible iff **(i)** no lateness occurs anywhere and **(ii)** cumulative load never exceeds (Q). This strict guard makes the search robust under tight windows and long services.

9. Complexity, Memory, and Runtime Budgets

- **Precompute**: (D\in\mathbb{R}^{(n+1)\times(n+1)}): (O(n^2)) time/space.
- **APCA**: (O(n\cdot K)) with bounded neighborhoods.
- **TFG & hypercliques:** neighborhood-restricted enumeration; practical (O(n\cdot K^2)).
- **DP split**: (O(n^2)) per tour (fast in practice with feasibility early-breaks).
- **RCF-LS:** time-capped (e.g., 3–5 min).
- **Memory**: always (O(n^2)) dominated by (D); fits in ≤ 12 GB for standard VRPTW sizes (e.g., Solomon/R-C/RC sets).

10. Parameters and Defaults (robust across datasets)

- Distance–time blend (\lambda\in[0.4,0.6]) (start 0.5).
- KNN neighborhood (K\in[8,16]) (default 10).
- Hyperclique sizes 3–5; cap stored cliques (e.g., 300).
- TW-2opt iterations (\approx 1.5!\times!10^3).
- DP evaluated on ≥ 5 tours (APCA, ATS, KCT, TW-NN, NN-distance).
- RCF-LS time budget 180–300 s; 2–5 seeds for robustness.
- Objective tie-breaks: (#routes, distance).

11. Reproducibility Protocol

- **Hardware**: single CPU core, ≤ 12 GB RAM, no GPU.
- **Seeds**: {7, 42, 99} (report best-of-k and mean±sd).
- **Budgets**: APCA ≤ 10 s, hypercliques ≤ 20 s, DP per tour ≤ 1 s, RCF-LS 180–300 s.
- **Outputs**: final routes (JSON/CSV), per-route and per-stop tables, route map, slack histogram, pipeline curves (initial→consolidated→final).

12. Why ALTAR-H is novel and useful

Unlike GA-centric or surrogate/XAI-guided sequencing frameworks, ALTAR-H **does not rely on learned surrogates**; instead, it **mines feasibility structure directly** (TFG + hypercliques) and couples it with **dual-potential attention** and **lexicographic DP + elimination LS** to prioritize **vehicle count** without sacrificing feasibility. The method remains **fully CPU-bound**, reproducible, and scales to real-world instances while producing **paper-ready** solution artifacts. The structured, two-stage organization (construction → improvement) follows best practice while being **algorithmically distinct** from GA/XAI hybrids.

Algorithm 1 (high-level pseudo-workflow)

1. Build (D), (\widehat{D}), temporal pressures (\widehat{\tau}); construct **APCA skeleton**.
2. Build **TFG**; mine **hypercliques** with at least one feasible order.
3. Generate **candidate tours** (APCA, ATS, KCT, TW-NN) + **TW-2opt**.
4. For each tour, run **Split-DP (lexicographic)**; pick best split; apply **DSTB consolidation** with clique priors.
5. Run **RCF-LS** (2-opt/Or-opt, relocate/swap, route elimination) under hard feasibility; stop at budget; return best.

This methodology section is intentionally concise yet complete so reviewers can reproduce the pipeline and understand **where** each improvement comes from (potentials, mined priors, DP split, elimination LS), **why** it is effective (lexicographic feasibility-first design), and **how** it runs on modest hardware.