# Assignment

## Comparing Methods on a Profit-Collecting VRPTW

Team Orienteering with Time Windows (TOPTW)

Applied AI: Academic perspectives: Combinatorial Optimisation Module

**Goal.** In this assignment you will compare *two* approaches among the techniques seen in class (Local Search, Simulated Annealing, Adaptive Large Neighbourhood Search, and, for small instances, exact methods). The emphasis is on the *process*: modelling choices, neighbourhood/operator design, parameter tuning, experimental method, and analysis of results. State-of-the-art optimality is *not* the goal.

## 1 Problem Summary (TOPTW)

You are given a depot and a set of customers with coordinates, demands, service times and *time windows*. Each customer $i$ yields a profit $p_i$ if served. The task is to select a subset of customers and partition them over at most $K$ vehicle routes (or pay a fixed cost per used vehicle) such that *total collected profit minus routing/vehicle costs* is maximised, while respecting capacity and time-window feasibility per route.

This problem is commonly known as the *Team Orienteering Problem with Time Windows* (TOPTW), and can be seen as a profit-collecting variant of the VRPTW. It is a good fit for ruin-and-recreate methods like ALNS and remains compatible with VRPTW neighbourhoods used in Local Search and SA.

## 2 Mathematical Model (Hard and Soft Time Windows)

We give a compact MIP with time and load propagation. This VRPTW-style flow with time ordering eliminates subtours.

### Data

- Depot 0; customers $V = \{1, \dots, n\}$; vehicles $k \in \{1, \dots, K\}$.
- Coordinates induce distances $d_{ij} \geq 0$ and travel times $t_{ij} \geq 0$ for $i, j \in \{0\} \cup V$, $i \neq j$ (assume $t_{ij} = d_{ij}$ unless stated).
- Demand $q_i \geq 0$, service time $s_i \geq 0$, time window $[a_i, b_i]$, profit $p_i \geq 0$ for $i \in V$.
- Vehicle capacity $Q > 0$; optional per-route duration limit $T_{\max} > 0$.
- Optional fixed cost $f \geq 0$ per used vehicle; lateness penalty $\beta \geq 0$ (only in the soft-TW variant).
- Big-$M$ constants $M_t, M_q$ sufficiently large to deactivate time/load propagation when an arc is unused.

## Decision Variables

- $x_{ijk} \in \{0,1\}$: arc $(i \to j)$ is used on route $k$ ($i \neq j$, $i, j \in \{0\} \cup V$).
- $y_i \in \{0,1\}$: customer $i$ is served.
- $u_k \in \{0,1\}$: vehicle $k$ is used.
- $t_i \geq 0$: service start time at node $i$ (one $t_0$ per route can be modelled via depot copies; below we use a single $t_0$ with big-$M$ activation).
- $w_i \geq 0$: vehicle load after serving $i$.
- *Soft-TW only:* $L_i \geq 0$ lateness at $i$ (penalised).

## Objective

**Hard time windows (no lateness):**

$$\max \quad \sum_{i \in V} p_i \, y_i \; - \; \sum_{k=1}^{K} \sum_{\substack{i,j \in \{0\} \cup V \\ i \neq j}} d_{ij} \, x_{ijk} \; - \; f \sum_{k=1}^{K} u_k. \tag{1}$$

**Soft time windows (lateness allowed, penalised):**

$$\max \quad \sum_{i \in V} p_i \, y_i \; - \; \sum_{k=1}^{K} \sum_{\substack{i,j \in \{0\} \cup V \\ i \neq j}} d_{ij} \, x_{ijk} \; - \; f \sum_{k=1}^{K} u_k \; - \; \beta \sum_{i \in V} L_i. \tag{2}$$

## Constraints

**Vehicle usage and depot degree (per vehicle):**

$$\sum_{j \in V} x_{0jk} = \sum_{i \in V} x_{i0k} = u_k \qquad \forall k = 1, \dots, K. \tag{3}$$

**Visit equals served (flow conservation over all vehicles):**

$$\sum_{j \in \{0\} \cup V} \sum_{k=1}^{K} x_{ijk} = \sum_{j \in \{0\} \cup V} \sum_{k=1}^{K} x_{jik} = y_i \qquad \forall i \in V. \tag{4}$$

**Capacity propagation:** For all $i \neq j$ in $\{0\} \cup V$,

$$w_j \; \geq \; w_i + q_j - M_q \left( 1 - \sum_{k=1}^{K} x_{ijk} \right), \qquad 0 \leq w_i \leq Q \, y_i \quad \forall i \in V. \tag{5}$$

**Time propagation:** For all $i \neq j$ in $\{0\} \cup V$,

$$t_j \; \geq \; t_i + s_i + t_{ij} - M_t \left( 1 - \sum_{k=1}^{K} x_{ijk} \right). \tag{6}$$

**Time windows:** For all $i \in V$,

$$a_i \le t_i \le b_i + M_t(1 - y_i) \quad \text{(hard-TW)} \tag{7}$$

or, in the soft-TW variant,

$$a_i \le t_i \le b_i + L_i + M_t(1 - y_i), \qquad L_i \ge 0. \tag{8}$$

**Route duration (optional, if modelled with depot times):**

$$t_0^{\text{return},k} \le t_0^{\text{start},k} + T_{\max} \qquad \forall k, \tag{9}$$

which can be implemented via depot copies per vehicle or accumulated travel/service-time counters.

*Remark.* The VRPTW-style time ordering together with degree/flow constraints removes subtours: time strictly progresses along used arcs, so cycles not connected to the depot are infeasible. Customer selection is governed by $y_i$. Calibrate $M_t, M_q$ tightly (e.g. using bounds derived from $b_i$ and $Q$) to improve LP relaxations.

## 3 Methods to Implement (Choose Two)

Pick any two of the following and ensure a fair experimental comparison. Make well consided choices!

**Exact (LP/MIP or B&B).** On small instances only (e.g. $n \le 30$–$40$).

**Local Search (LS).**

**Simulated Annealing (SA).**

**Adaptive Large Neighbourhood Search (ALNS).**

## 4 Experimental Protocol

**Instances.** Use the curated TOP/TOPTW sets derived from Solomon VRPTW (see Appendix A). For a soft-TW variant, use the same coordinates/TWs but allow lateness with penalty $\beta$ as in Appendix A.

**Difficulty tiers.**

- *Easy:* $n \approx 50$ (choose 3 instances; prefer class C or wider windows).
- *Hard:* $n \approx 100$ (choose 3 instances; include R/RC classes).

**Stochastic fairness.** For metaheuristics, run 5 independent seeds per instance. Report *mean*, *std*, and *best* objective.

**Budgets.** Same wall-clock budget per method and per instance (e.g. 2–5 minutes), identical initialisation policy across methods.

**Metrics.** Primary: assignment objective. Secondary: served customers, total distance, vehicles used, runtime.

**Tuning.** Use one development instance in each tier to tune a small parameter grid (e.g. SA cooling factor, ALNS scoring window, $\beta$). Fix parameters before final evaluation.

## 5 Deliverables

**D1. Reproducible Jupyter notebook** (Python) that:

- Parses instances (format in Appendix B).
- Implements your two chosen methods.
- Sets random seeds; logs runtime and objective.
- Produces summary tables/plots (per instance and aggregated).

**D2. Short report** (max 5 pages) covering:

- Modelling choices (hard vs soft TW, objective weights $f, \beta$).
- Neighbourhoods/operators and acceptance criteria.
- Tuning process (what you tried; how you selected final settings).
- Results with tables (*best/avg/std*), discussion of trade-offs.
- Recommendation: which method you would choose and why.

## 6 Practical Guidance

**Initialisation.** Greedy insertion by profit density $p_i/(d_{0i} + s_i)$ or regret-2 on the penalised objective.

**Feasibility.** Under hard TW, disallow infeasible moves. Under soft TW, allow temporary violations but penalise with $\beta L_i$ (time-warp idea).

**Relatedness (Shaw).** Combine Euclidean distance, TW overlap, and profit similarity to select related nodes for removal.

**Operator adaptation (ALNS).** Typical scores: +5 for global improvement, +2 for best-in-iteration, +1 for accepted; update weights every fixed window (e.g. every 100 iterations).

# A Benchmark Instances (Hard and Soft)

## Hard time windows (public TOPTW)

Use the widely adopted TOP/TOPTW instances derived from Solomon (classes C/R/RC; sizes around 50 and 100). Choose:

- **Easy tier (3 instances):** $n \approx 50$; include at least one class C (clustered) and one non-C.
- **Hard tier (3 instances):** $n \approx 100$; include R and/or RC classes (typically harder).

Many repositories list for each instance the best-known or optimal values (for small sizes). In your report, cite the source you used and indicate whether you compare against optimal values or best-known solutions where available.

## Soft-TW variant (derived)

To study penalty-based feasibility, derive a soft-TW version of any hard-TW instance by introducing nonnegative lateness variables $L_i$ and adding $\beta \sum_i L_i$ to the objective. Replace the constraint $t_i \leq b_i$ by $t_i \leq b_i + L_i$. Recommended starting range for $\beta$: 5 to 10 (tune on your development instance). Keep all other data unchanged.

*Note on difficulty.* Class C (clustered) with wider windows is often easier; R/RC and tighter windows are harder. Larger $n$ increases difficulty.

# B Simple File Format (CSV + JSON)

Provide each instance with two files.

### nodes.csv

```
id,type,x,y,profit,demand,service,tw_start,tw_end
0,depot,50.0,50.0,0,0,0,0,1e9
1,customer,12.3,77.1,84,3,10,120,240
...
```

### meta.json

```
{
  "K": 3,
  "Q": 100,
  "vehicle_fixed_cost": 200.0,
  "speed": 1.0,
  "distance_cost": 1.0,
  "T_max": null,
  "time_windows": "hard",          // "hard" or "soft"
  "lateness_penalty_beta": 8.0     // used only if "soft"
}
```

Compute $d_{ij}$ as Euclidean distance with double precision and set $t_{ij} = d_{ij}$ unless specified otherwise.

## C Academic Integrity & Reproducibility

Your notebook must allow full reproduction: list exact instance names, random seeds, parameter settings, library versions, and measured wall-clock limits. Any external code you adapt must be clearly attributed.

## D Suggested Reading (Optional)

Surveys and benchmark pages on Orienteering/TOPTW and Solomon-derived VRPTW data are widely available. Good starting points include a review of Orienteering variants, algorithmic studies on TOPTW (e.g. ILS/SA/ALNS), and the classic Solomon VRPTW description of instance classes (C/R/RC) and window patterns.