VRPTW TIG Challenge Description

TIG Labs | Aoibheann Murray

December 1, 2025

1 Problem Description and Formulation

The Vehicle Routing Problem with Time Windows (VRPTW) involves determining a set of cost-effective routes for a fleet of identical vehicles operating from a single depot to serve a geographically dispersed set of customers. Each vehicle has a fixed capacity and each customer has a known demand for goods and a defined time window during which service must begin. If a vehicle arrives before this time window, it must wait; if it arrives after, service is considered infeasible. The primary objective is to minimise the total distance the fleet must travel to deliver goods to all customers and return to the depot, such that:

- Each customer is visited by exactly one vehicle.
- The total demand serviced by each vehicle does not exceed its capacity.
- Each vehicle starts and ends its route at the depot.
- Service at each customer commences within the customer's defined time window.
- The number of vehicles utilised is less than a set fleet size.
- Vehicles wait if they arrive early, and service durations are accounted for within the schedule.

2 Random Instance Generation

The instance generation process is designed to produce scenarios that closely mirror established academic benchmark instances. It is crucial that these generated instances accurately reflect real-world conditions, as the game incentivises algorithms specifically optimised for these scenarios.

The instance generation process incorporates both random and clustered customer distributions, coupled with tight time window constraints and low vehicle capacity, characteristics defined as RC1 in Solomon's seminal paper on the VRPTW [1] and later utilised by Homberger and Gehring [2]. These instances serve as critical benchmarks for evaluating algorithms addressing the VRPTW. While benchmark instances typically vary in customer distribution (random, clustered, or mixed), time window tightness (loose or tight), and vehicle capacity (small or large), our focus is specifically on the RC1 type. This choice reflects more realistic and operationally challenging conditions, where tighter time constraints on vehicle arrivals require algorithms to achieve a delicate balance between route optimisation and precise scheduling. Such instances emphasise the need for greater accuracy in sequencing decisions, pushing algorithms to deliver both efficient routing and careful time management.

2.1 Instance Attributes

To align with almost all existing VRPTW benchmark instances, distances are two-dimensional Euclidean. Both the depot and customers are positioned at integer coordinates within a $[0, 1000] \times [0, 1000]$ grid. As described in the paper *New Benchmark Instances for CVRP* by Uchoa et al.[3], instances can be characterised by several attributes: the number of customers, depot positioning, customer positioning, demand distribution, and average route size. Below, we detail the choices made for these attributes for our instances.

Depot Positioning

Central (C) – the depot is positioned in the center of the grid, point $(500, 500)$.

Customer Positioning

Random-Clustered (RC) – half of the customers are positioned in clusters while the other half are randomly placed across the grid. Every customer, as well as the depot, is located at a unique point on the grid.

Customer positions are generated using a deterministic mixed random–clustered scheme, based off the acceptance probability model in Uchoa et. al. [3]. Clustering is achieved by sampling customers from truncated normal distributions centred on a small number of cluster seeds.
The generation process proceeds as follows:

- A number of cluster seeds k is sampled from a uniform distribution of [3,8]. Then each seed customer's position is sampled uniformly at random from the 1000×1000 grid.
- All subsequent customers (nodes $k + 1, \ldots, N$) are either assigned:
  1. Random placement (with probability 0.5): A position is drawn uniformly at random from the grid.
  2. Clustered placement (with probability 0.5): A cluster index $c \in \{1, \ldots, k\}$ is chosen uniformly at random, and the customer's coordinates are drawn independently from a truncated normal distribution centred at the seed's coordinates:
  $$[x \sim \mathcal{N}(x_c, \sigma^2), y \sim \mathcal{N}(y_c, \sigma^2)],$$
  where $[x_c, y_c]$ are the coordinates of the seed customer and $\sigma = 60$ was chosen based on experimental results to closely mirror the acceptance probability model in Uchoa et. al. [3]. The sampled values are truncated to [0,1000] and rounded to integers.
- Duplicate coordinates are not permitted. If a sampled location coincides with an already-used position, the sample is rejected and the process repeats for that node.

Overall, this yields a dataset composed of a mixture of uniformly distributed customers and locally clustered customers, where cluster density emerges naturally from the truncated Gaussian sampling rather than from the exponential decay attraction.

Demand Distribution

Small Values, Large Variance – demands from UD$[1, 35]$, where the range [1,35] was chosen such that the average demand is 17.5, yielding the target average route size.

Average Route Size

The average route size, r_{av}, is chosen to be in the range $\sim$ 11–12 customers per route, in line with the average route size chosen across instances in [3]:

$$r_{\text{av}} = \frac{\text{capacity}}{\text{avg\_demand}} = \frac{200}{17.5} = 11.43.$$

2.2 Time Window Generation

Depot Due Time

Based on experimentation and comparisons with the Solomon and Homberger instances[4], the following formula was selected for the depot due time:

$$l_0 = d_{0i_F} + (s_i + d_{\text{av}}) \times r_{\text{av}},$$

where l_0 is the depot due time, d_{0i_F} is the direct distance between the depot and the furthest customer i_F, s_i is the service time (with $s_i = 10$ for all i), and d_{av} is the average distance between customers.
The average customer distance d_{av} is derived from the mean distance between two points in a square of side length grid_size/2. As the depot is centered, we compute the average in a quarter of the grid using known formulas for random points in a square[5][6]:

$$d_{\text{av}} = \frac{1000}{2} \times 0.5214 = 260.7.$$

Therefore, the depot due time is

$$l_0 = d_{0i_F} + 3094.1.$$

Customer Due Times

The due times of the time windows are determined differently depending on whether customers are randomly distributed or clustered.
Initially, every customer is assigned a due time drawn uniformly from the interval

$$[d_{0i},\ l_0 - d_{0i} - s_i],$$

where d_{0i} is the distance from the depot to customer i, l_0 is the depot's due time, and s_i is the customer's service time.
This ensures that each time window is feasible: a vehicle can leave the depot, reach the customer, perform the service, and return on time.

Clustered Customers For clustered customers, the due times are adjusted to be closer to their respective seed customer's due time. This is achieved by recalculating each clustered customer's due time as the average of its original due time and that of its seed, truncated to the original bounds.

Ready Times

The depot's ready time is set to zero. The ready times for a subset of customers is defined as their due time minus a randomly selected time window width chosen from a uniform distribution UD$[10, 60]$. The density parameter, currently set at 50%, determines the proportion of customers with non-zero ready times. This yields variability in time window widths, producing both tight and loose windows.

2.3 Decisions on Conventions

1. Euclidean distances are rounded to the nearest integer, as in [3].
2. The number of routes in a solution is not fixed, but an upper bound will be set based on the baseline solution.

3 Two-phase Verification: Baseline Calculation

As part of TIG's verification process, to prevent a malicious attack and to assess the quality of a solution, we have a two-phase verification process.
All instances on TIG are randomly generated client side, meaning we do not know the optimal solution. In order to gauge the quality of a solution, we run a 'baseline' algorithm, which returns a reference solution we can compare against.

3.1 Two-tier Verification

Tier 1 — Proof-of-work verification

Verify the submitted solution is at least as good as the cheap baseline (e.g., pure greedy). This tier protects the network from Distributed Denial-of-Service (DDoS) attacks.

Tier 2 — Quality Measurement

Compute the solution's quality by comparing it against the solution found by a sophisticated baseline.

The baseline calculation gives a reference performance metric for each instance. A key feature of the baseline algorithm is stability, i.e, the variance of the baseline solution from the optimal value should be low.

3.2 Proof-of-Work Baseline

Solomon's I1 heuristic is used of the 'cheap' baseline for proof-of-work purposes. Solomon's I1 heuristic is a widely recognised constructive approach for solving the VRPTW. It incrementally constructs routes by inserting customers into positions that minimise an insertion cost, while satisfying vehicle capacity and time window constraints.

1. **Initialisation:** Start with an empty route and select an initial customer to serve.
2. **Insertion Process:** For each unrouted customer:
   - Evaluate all feasible positions in the current route.
   - Compute the **insertion cost** at each position, considering both distance and time adjustments.
   - Select the position that minimises the cost while maintaining feasibility (capacity and time window constraints).
3. **Route Completion:** Insert the chosen customer into the route. Repeat until no more customers can be added.
4. **Open a New Route:** If unrouted customers remain, open a new route and repeat the process until all customers are routed.

The insertion cost is computed using two levels of cost functions:

First-Level Cost (C_1)

The first-level cost prioritizes minimising added distance and time adjustments. For a candidate customer u inserted between two consecutive nodes i_{p-1} and i_p, the cost function is:

$$C_1(i_{p-1}, u, i_p) = a_1[d(i_{p-1}, u) + d(u, i_p) - \mu d(i_{p-1}, i_p)] + a_2(b_{j,u} - b_j),$$

where:

- $d(x, y)$ is the distance between nodes x and y,
- $b_{j,u}$ and b_j represent service start times,
- a_1, a_2, μ are parameters that balance distance and time impacts.

Second-Level Cost (C_2)

The second-level cost adjusts for the proximity of the candidate customer to the depot and is defined as:

$$C_2(i_{p-1}, u, i_p) = \lambda d(0, u) - C_1(i_{p-1}, u, i_p),$$

where λ weights the depot's distance influence.

4 Difficulty parameter: Size

Within each challenge, there are various challenges tracks. These can range over instance size and/or type. Currently, the challenge supports varying sizes of VRPTW instances as defined by Solomon[7]:

- 500
- 600
- 700
- 800
- 900
- 1000

For this challenge, as number of customers, N, increases, the problem becomes significantly more complex. A larger customer set expands the solution space and increases the number of feasible routes, making it more challenging to identify the optimal solution. The growing complexity with higher N tests an algorithm's ability to scale efficiently while maintaining performance.

5 Quality

The parameter that measures the quality of a given solution is known as 'better_than_baseline'. It represents an algorithm's performance, by requiring solutions to achieve an improvement over the sophisticated baseline algorithm. Gaining a high quality score will result in more rewards, creating a consistent demand for algorithms to deliver higher-quality solutions.

References

[1] Marius M. Solomon. "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints". In: *Oper. Res.* 35 (1987), pp. 254–265. URL: `https://api.semanticscholar.org/CorpusID:15346313`.

[2] Jörg Homberger and Hermann Gehring. "Two Evolutionary Metaheuristics For The Vehicle Routing Problem With Time Windows". In: *Infor* 37 (1999), pp. 297–318. URL: `https://api.semanticscholar.org/CorpusID:14190711`.

[3] Eduardo Uchoa et al. "New benchmark instances for the Capacitated Vehicle Routing Problem". In: *European Journal of Operational Research* 257.3 (2017), pp. 845–858. ISSN: 0377-2217. DOI: `https://doi.org/10.1016/j.ejor.2016.08.012`. URL: `https://www.sciencedirect.com/science/article/pii/S0377221716306270`.

[4] SINTEF. *Vehicle Routing Problem with Time Windows (VRPTW)*. `http://www.sintef.no/vrptw`. Accessed: 2024-12-05. Accessed 2024.

[5] Eric W. Weisstein. *Square Line Picking*. Accessed: 2024-12-05. n.d. URL: `https://mathworld.wolfram.com/SquareLinePicking.html`.

[6] *Mean line segment length*. Accessed: 2024-12-05. n.d. URL: `https://en.wikipedia.org/wiki/Mean_line_segment_length#cite_note-:0-3`.

[7] Marius M. Solomon. “Algorithms for the vehicle routing and scheduling problems with time window constraints”. In: *Operations Research* 35.2 (1987), pp. 254–265.