SMIO–Hexaly Location Routing Challenge Monterrey 2026

Problem Specification

Version 1.0

Prepared by: Eduardo Salazar
Board Member, SMIO

On behalf of the SMIO Board of Directors

April 2026

Contents

1. Problem description

1.1. The Capacitated Location Routing Problem (CLRP)

We consider the **Capacitated Location Routing Problem (CLRP)**, an extension of the classical problem that combines strategic facility location decisions with operational vehicle routing decisions.

A set of potential depot locations and a set of customers with known demands are given. A fleet of identical vehicles is available to serve customers via routes that start and end at an open depot. The goal is to simultaneously decide which depots to open, assign customers to open depots, and design vehicle routes, minimizing total cost.

This formulation extends the classical CLRP with two operational enrichments: a fixed cost incurred each time a vehicle is dispatched and a maximum number of vehicles that each depot can accommodate.

The challenge website will include a sample instance, a feasible solution format, and the solution validator code so that participants can understand what constitutes a feasible solution and how the objective function is computed.

1.2. Industrial relevance

The CLRP is directly relevant to Mexican industry. Companies such as OXXO, Bimbo, and FEMSA routinely face decisions about where to place distribution centers while simultaneously optimizing delivery routes. Framing the challenge around this problem resonates with both academic and industry audiences.

From an algorithmic standpoint, the LRP sits at a difficulty sweet spot: it combines strategic location decisions with operational routing, meaning no single algorithmic paradigm dominates. This opens the competition to teams using metaheuristics, decomposition methods, branch-and-price, matheuristics, machine learning, or hybrid approaches.

There is also less benchmark saturation compared to the CVRP, so the challenge fills a genuine gap in the community and any results carry higher novelty.

2. Input data

2.1. Sets

- $I = \{1, \ldots, m\}$: set of potential depot locations.
- $J = \{1, \ldots, n\}$: set of customers.

2.2. Depot parameters

For each depot $i \in I$:

- (x_i^d, y_i^d): coordinates of depot i.
- f_i: fixed cost of opening depot i.
- W_i: capacity of depot i (maximum total demand it can serve).
- V_i: maximum number of vehicles that can operate from depot i.

2.3. Customer parameters

For each customer $j \in J$:

- (x_j^c, y_j^c): coordinates of customer j.
- q_j: demand of customer j (with $q_j > 0$).

2.4. Vehicle and routing parameters

- Q: vehicle capacity (identical for all vehicles).
- g: fixed cost per dispatched route.

2.5. Distances

d_{ab}: travel distance between any two nodes $a, b \in I \cup J$.

For **Euclidean instances** (small and medium scales), pairwise distances are computed as:

$$d_{ab} = \text{round}\left(\sqrt{(x_a - x_b)^2 + (y_a - y_b)^2},\ 1\right)$$

The rounding to one decimal place is applied *pairwise, before summation*. The cost of a route is the sum of these rounded edge costs; no further rounding is applied at the route or total level. This rule is normative and enforced by the official verifier.

For **explicit instances** (large scale), the full distance matrix is provided directly in the instance file. These distances are derived from real road-network routing on the Monterrey Metropolitan Area and may be asymmetric (i.e., $d_{ab} \neq d_{ba}$ in general). The verifier does not assume the triangle inequality on explicit distances.

3. Decisions and constraints

3.1. Decisions

1. **Depot opening**: which depots $i \in I$ to open.
2. **Customer assignment**: each customer $j \in J$ must be assigned to exactly one open depot.
3. **Route design**: for each open depot, design a set of vehicle routes serving all customers assigned to that depot. Each route starts and ends at its assigned depot.

3.2. Constraints

1. **Customer service**: every customer must be visited exactly once by exactly one route.
2. **Depot assignment**: each route operates from exactly one open depot, and every customer on that route is assigned to that depot.
3. **Vehicle capacity**: the total demand of customers on any route must not exceed the vehicle capacity Q:

$$\sum_{j \in R_k} q_j \leq Q \qquad \forall \text{ route } k$$

4. **Depot capacity**: the total demand of all customers assigned to depot i must not exceed its capacity W_i:
$$\sum_{j \in S_i} q_j \leq W_i \qquad \forall i \in I \text{ with } y_i = 1$$
where S_i is the set of customers assigned to depot i and $y_i = 1$ indicates that depot i is open.
5. **Vehicle limit**: the number of routes operating from depot i must not exceed V_i:
$$|\{k : \text{route } k \text{ operates from depot } i\}| \leq V_i \qquad \forall i \in I \text{ with } y_i = 1$$

3.3. Objective function

Minimize total cost:

$$\min \sum_{i \in I} f_i \, y_i \; + \; g \cdot |\mathcal{K}| \; + \; \sum_{k \in \mathcal{K}} \text{dist}(R_k)$$

where:

- The first term is the total depot opening cost.
- The second term is the total fixed cost for dispatching routes (g per route, with $\mathcal{K}$ the set of all routes).
- The third term is the total travel distance, with $\text{dist}(R_k)$ the distance of route k (depot $\rightarrow$ first customer $\rightarrow \cdots \rightarrow$ last customer $\rightarrow$ depot).

4. File formats

4.1. Instance format

Instance files use plain text format, delimited by whitespace. Lines beginning with `#` are comments and are ignored by parsers; they are used to carry provenance metadata (generator seed, version, OSM extract date).

```
# seed=<int>
# generator_version=<semver>
# osm_extract=<YYYY-MM-DD>          (only for FULL_MATRIX instances)
NAME : <instance_name>
CUSTOMERS : <n>
DEPOTS : <m>
VEHICLE_CAPACITY : <Q>
ROUTE_FIXED_COST : <g>
DISTANCE_FORMAT : COORDS | FULL_MATRIX
DEPOT_SECTION
<depot_id> <x> <y> <opening_cost> <capacity> <max_vehicles>
...
CUSTOMER_SECTION
<customer_id> <x> <y> <demand>
...
DISTANCE_SECTION
<full (m+n) x (m+n) distance matrix, row by row>
EOF
```

Notes:

- The `DISTANCE_SECTION` is only present when `DISTANCE_FORMAT: FULL_MATRIX`.
- When `DISTANCE_FORMAT: COORDS`, distances are Euclidean (rounded pairwise to one decimal place before summation).
- Node indices in the distance matrix follow the order: depots $1, \ldots, m$, then customers $m+1, \ldots, m+n$.
- All values are non-negative. Demands are strictly positive integers. Distances and costs are non-negative reals.

4.2. Solution format

```
# instance=<instance_name>
COST : <total_cost>
DEPOTS_OPENED : <number_of_open_depots>
ROUTES : <total_number_of_routes>
DEPOT <depot_id>
  ROUTE : <customer_id_1> <customer_id_2> ... <customer_id_last>
  ROUTE : <customer_id_1> <customer_id_2> ...
DEPOT <depot_id>
  ROUTE : <customer_id_1> ...
...
EOF
```

Notes:

- Each `DEPOT` block lists the routes operating from that depot.
- Each `ROUTE` line lists customer IDs in visitation order (the depot is implicit at the start and end).
- The `COST` value must match the objective computed from the solution within a tolerance of 10^{-4}.
- Each customer must appear exactly once across all routes.
- Empty routes are rejected. Depot blocks with no routes should be omitted.

4.3. Solution verification

A submitted solution is **feasible** if and only if all of the following conditions hold:

1. Each customer appears in exactly one route.
2. Each route is assigned to an open depot.
3. The total demand on each route does not exceed Q.
4. The total demand assigned to each open depot does not exceed W_i.
5. The number of routes at each open depot does not exceed V_i.
6. The reported cost matches the recomputed cost within tolerance 10^{-4}.

An infeasible solution is rejected and does not update the leaderboard.

5. Instance design

Instances will be generated using an open-source Python generator and will cover the following dimensions:

Dimension	Range
Customers (n)	200 – 3,000
Depots (m)	10 – 50
Number of instances	30
Distance type	Euclidean (small/medium); Monterrey road-network, asymmetric (large)

5.1. Structural variation

Instances will vary along the following axes to prevent over-specialization:

- **Customer distribution**: uniform random, clustered, mixed (clusters + uniform).
- **Depot positioning**: grid, random, co-located with customer clusters, peripheral.
- **Demand distribution**: uniform, bimodal (light and heavy customers), proportional to cluster size.
- **Capacity tightness**: loose (ample depot and vehicle capacity), moderate, tight (capacity is an active constraint).
- **Vehicle limit tightness**: loose (limits rarely active) to tight (forces depot opening).
- **Route cost ratio**: low g (many short routes preferred) to high g (fewer long routes preferred).

For the large-scale instances, customer and depot locations are sampled from real polygons in the Monterrey Metropolitan Area (residential and commercial zones for customers; industrial zones for depots), and distances are computed via self-hosted OSRM routing on a Geofabrik Nuevo León extract. This anchors the challenge in genuinely Mexican industrial geography and strengthens the resonance with the OXXO / Bimbo / FEMSA framing.

The exact distribution of instances across these axes will be finalized before the competition launch and published in the generator's design-matrix configuration.

6. Competition format

6.1. Tracks

The challenge features two tracks:

a) **Undergraduate Track**: teams of up to 3 current undergraduate students plus 1 guide (defined as anyone who is not a current undergraduate student). This track aligns with SMIO's educational mission and encourages mentorship.

b) **Open Track**: teams of up to 3 participants, no restrictions on profile. Open to students, researchers, and industry professionals.

At least half of the team members must hold a current SMIO/ALIO membership.

6.2. Scoring and leaderboard

6.2.1. Hexaly baseline and participant BKS

Before the competition opens, Hexaly Optimizer runs on all instances to produce a **baseline solution** for each instance. These baselines are published as a reference but **do not participate in the leaderboard**. The leaderboard tracks only solutions submitted by participant teams.

This means that the Best Known Solution (BKS) on the leaderboard is always held by a participating team, and the lead-time scoring system (described below) always produces a meaningful ranking regardless of whether any team improves upon Hexaly's baseline.

At the end of the competition, results will be presented alongside the Hexaly baseline, providing a clear comparison: for each instance, it will be reported whether participants matched, improved, or fell short of the Hexaly solution. This preserves the compelling narrative of the challenge — the community benchmarking itself against a strong industrial solver — while ensuring the competition always has a well-defined winner.

6.2.2. Lead-time scoring

Each instance has a participant BKS that evolves during the competition, starting with the first feasible submission received.

For each instance ℓ and each team t, let $\Delta_{\ell,t}$ be the cumulative time, measured in days, during which team t holds the participant BKS of instance ℓ. The measurement is *sliding*, with second-level precision: a team's timer starts at the exact server-received timestamp of the submission that establishes their BKS, and stops at the exact instant another team supersedes it or the competition closes. Thirty-six hours of lead, for instance, counts as 1.5 days. In the event of a tie on objective value, the incumbent team retains the BKS.

Let $b_{\ell,t} = 1$ if team t holds the final participant BKS of instance ℓ at the close of the competition, and 0 otherwise.

The score of team t is:

$$S_t = \sum_{\ell} \Delta_{\ell,t} \; + \; B \sum_{\ell} b_{\ell,t}$$

where B is a bonus (in equivalent days) for holding the final BKS. The team with the highest total score wins in each track. The value of B is set to 3 days.

6.2.3. Leaderboards

Four public views will be maintained:

1. **Per-instance leaderboard**: for each instance, the current participant BKS cost, the team holding it, and the timestamp at which it was established.
2. **Global leaderboard**: the aggregate score S_t of each team, updated in real time.
3. **Hexaly comparison**: for each instance, the Hexaly baseline cost alongside the current participant BKS, with the percentage gap between them.
4. **Per-instance detail**: for each instance, its basic statistics (scale, number of customers and depots) and the complete chronological evolution of its participant BKS during the window: which team held it, at what cost, and between which instants. The solution files themselves

and the per-submission metadata (computation time, environment) remain withheld until the close of the competition.

The current BKS cost and the team holding it are public in real time. The actual solution files are withheld until the end of the competition, preventing teams from locally improving each other's submissions.

When a submission establishes a new BKS, the submitting team is prompted (after verification) to provide additional metadata on that specific submission: wall-clock computation time in seconds, and a brief description of the computational environment (software versions, CPU, RAM). This metadata is recorded alongside the solution in the benchmark library.

6.3. Method description

Each team submits a concise PDF describing their approach at the time of team registration. The document covers: problem reformulation (if any), method overview, tools and solvers, hardware, and computational budget. The description is reviewed by the SMIO Board as part of the registration approval process. At the end of the competition, the method descriptions of participating teams are published alongside their BKS-establishing solutions in the benchmark library, turning the artifact from a dataset into a citable collection of approaches.