A Two-Staged Hybrid Genetic Search with Route-Clustering Intensification for the CVRPLIB BKS Challenge

MAMUT team: Adrien Pichon, Alexandru Olteanu, Christine Solnon, Marc Sevaux

January 4, 2026

Abstract

This report describes a CVRP solver developed with the objective of producing competitive best-known solutions (BKS) for standard benchmarks, in the spirit of the CVRPLIB BKS challenge. The method builds on the Hybrid Genetic Search (HGS) framework of Vidal et al. and adds three practical components aimed at improving time-to-quality and robustness across instance sizes: a diversified population initialization scheme, a route-clustering intensification operator solved by lightweight sub-HGS runs in parallel, and an extended local search with adaptive cost controls. The resulting solver preserves the overall HGS lifecycle while improving exploration and regional intensification on large instances.

1 Motivation and Objectives

The CVRPLIB BKS challenge encourages the continuous improvement of best-known solutions on widely used CVRP instances. In practice, success requires both high solution quality and consistent behavior across heterogeneous instance sizes and structures. This project targets that goal by retaining a strong HGS backbone while adding mechanisms that (i) reach good solutions earlier, (ii) intensify search in meaningful regions without exploding runtime, and (iii) use parallel hardware effectively when available.

2 Baseline: Hybrid Genetic Search (HGS)

The solver is based on the Hybrid Genetic Search paradigm introduced by Vidal [2] and later consolidated in an extended view of HGS components for VRP [1]. We keep the main design choices and operator sequence. Solutions are encoded as *giant tours* and decoded by a *Split* procedure into routes. A dedicated education step (local search) improves individuals, feasible and infeasible subpopulations are maintained with diversity control, and penalties are dynamically adapted to balance feasibility and exploration.

The developments described below do not replace this architecture. Instead, they act as additional diversification/intensification layers that can be turned on selectively and tuned with explicit parameters.

3 Diversified Initialization: Better Time-to-Quality

On large instances, a purely random initial population can be diverse but needs many generations before reaching a quality level where intensification becomes productive. To mitigate this warm-up cost while preserving diversity, the initial population is built from a mixture of constructions.

A first fraction of the population is generated exactly as in the classical HGS, i.e., as random giant tours. The second fraction uses a randomized variant of a savings-based construction: savings values are perturbed through randomized weights and tie-breaking so that multiple runs yield structurally different (yet typically good) solutions. The last fraction uses a sweep-based construction: customers are ordered by their polar angle around the depot, but the angles are deliberately perturbed or locally shuffled so that the sweep produces different route skeletons across individuals.

All constructed tours are passed through the same pipeline as standard individuals (Split and education, with repair if needed), so the remainder of population management and selection remains unchanged. Empirically, this approach yields an initial population that is simultaneously more exploitable (lower initial costs) and still diversified (different route structures).

4 Route-Clustering Intensification with Parallel Sub-Solvers

4.1 Rationale

Many improvements in large CVRP instances are essentially regional: routes operating in the same geographic area benefit from re-optimization together, while interactions between far-apart routes are less frequent. A global local search can handle these interactions, but it becomes expensive to apply repeatedly if the search space is large. To address this, we introduce a periodic, solution-dependent intensification operator that temporarily decomposes a solution into smaller subproblems.

4.2 Centroid features and K-means clustering

Given a decoded solution (a set of routes), we compute a route-level feature: the centroid (center of gravity) of each route based on its customers,

$$C_x = \frac{1}{|R|}\sum_{i\in R} x_i, \qquad C_y = \frac{1}{|R|}\sum_{i\in R} y_i.$$

Routes are then clustered using standard K-means in the two-dimensional centroid space. A fixed K is avoided: we compute it dynamically from the number of routes, using a target number of routes per cluster (typically 3–5). This keeps clusters sufficiently small to be solved quickly, while still allowing meaningful rearrangements inside each region.

4.3 Sub-instance generation and lightweight Sub-HGS

Each cluster induces a sub-instance formed by the union of customers contained in the cluster routes, plus the original depot. A local indexing map (global $\leftrightarrow$ local) is maintained so that sub-solutions can be translated back correctly.

Each sub-instance is solved by a lightweight configuration of the same HGS machinery: small populations, short time limits, and simplified stopping criteria. Importantly, clustering is disabled inside sub-solves to avoid recursive nesting. This yields a "nested" intensification effect without the prohibitive cost of running full HGS inside full HGS.

4.4 Parallel execution and reassembly

Sub-instances are independent, so they are solved in parallel. The operator then reassembles the optimized routes from all clusters into a global solution and applies one global education pass. This final local search is essential: it allows boundary corrections between clusters and reintroduces cross-cluster moves that were not allowed during sub-solves.

From an accounting standpoint, the time attributed to this operator is the wall-clock duration of the parallel phase (not the sum of per-cluster runtimes), ensuring that the operator scales with the available cores.

4.5 When to apply

Because this operator can still be expensive, it is not applied to every offspring. Instead, it is used periodically and/or on promising individuals, and only on instances above a minimum size threshold. The sub-solver time budget is also scaled based on cluster size and remaining global time.

5 Local Search Extensions with Cost Control

We extend the education phase while keeping it computationally reasonable. The guiding idea is to enrich the neighborhood landscape but compensate for this added breadth with filtering and adaptive budgeting.

Routes maintain additional metadata such as slack (capacity/duration), modification counters, and a simple priority score. This score emphasizes routes under penalty pressure or those that have stagnated, and it is used to decide where to spend local-search effort. In stable routes, the algorithm explores only a limited prefix of the correlated-neighbor lists; during stagnation, the exploration budget is progressively expanded to restore full neighborhood coverage before declaring convergence.

The move set is enriched with bounded-cost operators chosen to fit the granular-search philosophy of HGS. In addition to the standard relocate/exchange/2-opt style operations from the original implementation, we include short Or-opt relocations (segments of length 3–4), restricted cross-exchange between selected route pairs, and shallow ejection-chain style sequences (very small depth). Each move is protected by inexpensive feasibility/slack checks to avoid evaluating obviously infeasible insertions.

Finally, when improvement stalls, a small destroy/repair "kick" is applied. A few customers (or short segments) are removed from high-priority routes and greedily reinserted. This perturbation is throttled by a cooldown so it does not dominate runtime.

6 Two-Stage Parameter Optimization

Performance depends heavily on parameters and the diversity of instance characteristics. To tune robustly for the CVRPLIB setting, we implemented a two-stage tuning procedure.

In the first stage, a coarse exploration is performed via Latin-hypercube sampling over key knobs: granularity, population sizes, penalty update factors, clustering interval and sub-solver time, neighbor budgets, and perturbation intensity. Configurations are evaluated on a tiered training set drawn from the `X` and `XL` families, and each configuration is run for multiple random seeds. The run logs record best cost and elapsed time to produce a cost–runtime trade-off view.

In the second stage, we fit a surrogate model per tier (e.g., random forests or Gaussian processes) and run Bayesian optimization under runtime constraints to refine the search in regions that look promising. The final selection uses Pareto efficiency and robustness criteria: profiles are retained if they remain within a target gap across tiers, and then re-tested on unseen instances, including very large cases up to `n=10001`.

7 Conclusion

The solver developed in this project is a direct descendant of the Hybrid Genetic Search framework of Vidal et al. [2, 1], augmented with components that target the practical requirements of the CVRPLIB BKS challenge. Diversified initialization improves time-to-quality, route-level K-means clustering enables scalable regional intensification through parallel lightweight sub-solves, and the education phase is strengthened through additional neighborhoods combined with adaptive filters and bounded perturbations. A two-stage, tier-aware tuning workflow supports robust parameter choices that generalize across instance sizes, from the `X` set to very large `XL` benchmarks.

References

[1] T. Vidal. Hybrid genetic search for the CVRP: open-source implementation and swap* neighborhood. *CoRR*, abs/2012.10384, 2020. URL `https://arxiv.org/abs/2012.10384`.

[2] T. Vidal, M. Gendreau, N. Lahrichi, and W. Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624, 2012. doi: 10.1287/opre.1120.1048.