A Three-Tier Cooperative Search Framework with LLM-Evolved Components for the CVRPLib BKS Challenge

CVRPLib BKS Challenge — Technical Report Document

1 Team Information

Team Name: OptVerse-CityU

Members: Zhiwu An[1], Xin Chen[2], Qinglong Hu[2], Fei Liu[2], Mahdi Mostajabdaveh[1], Zidong Wang[2], Shunyu Yao[2], Kefeng Zheng[2], Zirui Zhou[1], Xialiang Tong[1], Mingxuan Yuan[1], Qingfu Zhang[2]

Affiliations: [1]Huawei Technologies, [2]City University of Hong Kong

Contact: anzhiwu1@huawei.com, xchen3252-c@my.cityu.edu.hk, qinglhu2-c@my.cityu.edu.hk, fliu36-c@my.cityu.edu.hk, mahdi.mostajabdaveh1@huawei.com, zidowang@cityu.edu.hk, shunyuyao8-c@my.cityu.edu.hk, kefezheng2-c@my.cityu.edu.hk, zirui.zhou@huawei.com, tongxialiang@huawei.com, yuan.mingxuan@huawei.com, qingfu.zhang@cityu.edu.hk

** Team members are listed in alphabetical order by surname.*

Abstract

We present a hybrid metaheuristic framework that extends AILS-II [1] with two complementary enhancements: (i) **LLM-guided component design** using Evolution of Heuristics (EoH) to automatically evolve improved diversification–intensification mechanisms, and (ii) **a three-tier cooperative search architecture** that balances global exploration, warm-start intensification, and greedy exploitation across the 30-day competition horizon. Our approach specifically targets the late-stage search regime where marginal improvements over already high-quality solutions are most difficult to achieve.

2 Methodology

2.1 Base Algorithm: AILS-II

We build upon AILS-II [1], an iterated local search variant with adaptive diversity control. Three parameters govern the diversification–intensification trade-off:

- **Stopping time:** Controls convergence speed. A longer stopping time promotes global exploration, while a shorter stopping time drives rapid local convergence. This provides a natural mechanism to trade off exploration and exploitation.
- **Perturbation degree (ω):** Dynamically adjusted with frequency γ based on the ideal distance between the reference solution and the post-local-search solution. The ideal distance decreases over time, gradually shifting the search from diversification to intensification.
- **Acceptance criterion:** A threshold-based criterion that accepts a solution if its cost satisfies $\theta = f^{\text{best}} + \eta(\bar{f} - f^{\text{best}})$, where f^{best} is the best solution found in the last γ iterations and $\bar{f}$ is the average local search quality. The relaxation parameter $\eta \in [\eta_{\min}, \eta_{\max}]$ starts permissive ($\eta_{\max}$) and converges to restrictive ($\eta_{\min}$) over time.

2.2 Main Framework

Motivation. Since the competition spans 30 days with cumulative scoring, no single algorithm configuration dominates across all stages. We therefore design a three-tier framework that assigns distinct roles to algorithm variants based on the competition phase. Figure 1 illustrates the overall architecture.

- **Global Exploration.** These methods run AILS-II independently with long stopping times (1–20 days). They write their best solutions to the shared database but never read from it. The rationale is twofold: (a) long-horizon AILS-II achieves the best asymptotic solution quality due to its simulated-annealing-like acceptance criterion; (b) allowing these runs to read from the shared pool risks premature convergence to local optima already found by other workers.
- **Cyclic Warm Start.** These are AILS-II variants with medium stopping times (1–5 days) that run repeatedly in cycles. At the start of each cycle, they initialize from the current best solution in the shared database, then contribute any improvements back during execution. This bidirectional interaction enables fine-grained local intensification (due to their shorter run-time) on top of the high-quality solutions produced by Global Exploration.
- **Greedy Exploitation.** These algorithms, including HGS, FILO2, and AILS-II variants with restrictive acceptance criteria, read from and write to the shared database immediately upon finding a new best solution. They serve two purposes: (a) rapid convergence in the early competition stage to establish a strong leaderboard position; (b) diverse neighborhood structures

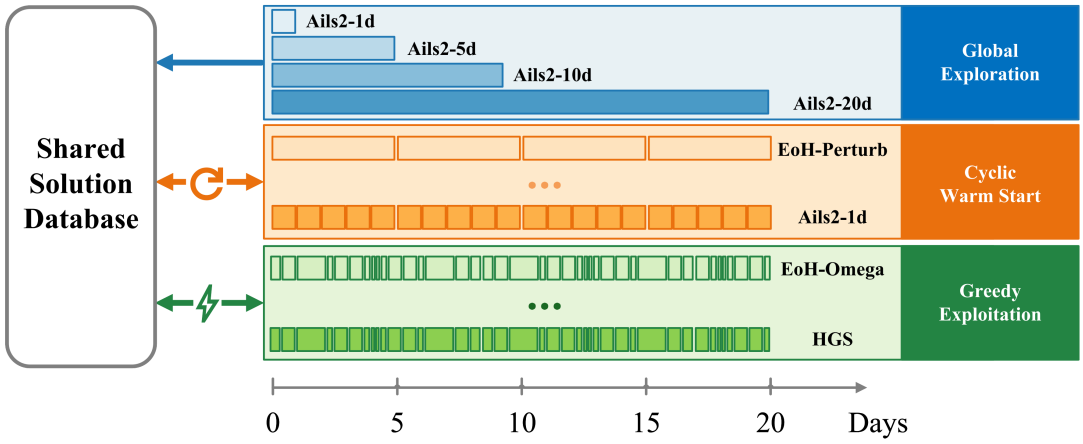

Figure 1: Overview of the three-tier cooperative search framework.

(e.g., population-based search in HGS) that help escape local optima encountered by other tiers.

- **Shared Database.** For each instance, the parallel framework employs a SQLite-based shared database as the central communication medium among all algorithm processes. The shared database maintains three categories of information: (a) Global Best Solution: A single record that always reflects the best solution found so far across all running algorithms. (b) Solution History: A cumulative log of every improving solution submitted by any algorithm throughout the entire run. (c): Improvement Events: A record of each occasion when a new global best was established. For synchronization mechanism, concurrent write access is managed through SQLite's Write-Ahead Logging (WAL) mode. The main coordinating process periodically polls the shared database at a fixed time interval to check whether the global best has improved. Upon detecting an improvement, it identifies which algorithms are no longer competitive in the Greedy Exploitation tier, terminates them, and restarts them using the newly found best solution as their initial seed.

Dynamic resource reallocation. As Global Exploration runs complete over time, the freed computational resources are redirected to additional Cyclic Warm Start instances. This naturally shifts the overall search from exploration-dominant in the early stage to intensification-dominant in the late stage. As the competition progresses and scores begin to converge, further improvements become increasingly elusive; to counteract this, we manually deploy additional, longer-running refinement processes to exhaustively exploit high-potential regions.

2.3 LLM-Guided Component Design via EoH

Motivation. Evolution of Heuristics (EoH) [2] is a framework for automated algorithm design that uses Large Language Models to generate, mutate, and recombine algorithmic components. We apply EoH to evolve three core components of AILS-II on the CVRPLib XLTEST dataset prior to the competition, with the goal of improving late-stage convergence behavior.

Design process.

- **Training instances:** We selected instances with large customer counts and varied distribution parameters from the CVRPLib XLTEST to encourage generalization.
- **Warm-start evaluation:** To focus evolution on the late-stage regime, initial solutions are generated by running AILS-II for 5 days; evolved components then continue the search for 1 hour. This ensures fitness reflects improvement over already high-quality solutions.
- **Evolutionary operators:** The LLM generates novel functions (generation), modifies high-performing functions (mutation), and combines elements from multiple successful functions (crossover). Selection is based on fitness ranking over the candidate population.
- **Hyperparameters:** For the design of a single operator, we conducted approximately 13 generations with a population size of 16 per generation, totaling roughly 200 samples. The evaluation time for each sample was one hour, with Gemini-2.5-pro serving as the base Large Language Model.

Evolved components.

- **Acceptance criterion:** Replaces the default threshold rule with an automatically designed criterion for deciding whether to accept a new reference solution. Two variants are evolved: one for standard instances implements *Threshold Accepting with*

Adaptive Patience (TAAP), which tracks the last accepted solution cost and widens the acceptance window upon stagnation while tightening it upon improvement; the other for large instances implements *Strategic Oscillation with Adaptive Phasing* (SOAP), which alternates explicitly between a greedy intensification phase and a threshold-based diversification phase.

- **Perturbation control (*OmegaAdjustment*):** Replaces the hand-crafted ω update rule with an evolved function that adapts perturbation degree based on search state. The evolved rule uses a three-state finite-state machine: fine-grained PI-like adjustments under normal conditions, a large corrective step when the perturbation degree deviates significantly from the ideal, and a forced boost when the search stagnates completely.
- **Perturbation approach (ruin operator):** Extends the original *Sequential* and *Concentric* removal strategies with a novel EoH-designed operator for selecting which customers to remove from the current solution. The evolved *Route-Based Split Ruin* operator selects a seed customer and expands the removal set alternately forward and backward along the route, filling any shortfall with customers from other routes. This creates a localized, contiguous disruption that is structurally distinct from the existing operators, thereby diversifying the perturbation landscape.

The EoH-designed components are deployed in both the Greedy Exploitation tier (for rapid early convergence) and the Cyclic Warm Start tier (for diverse search paradigms that help escape local optima in later stages).

2.4 Method Summary

Table 1 lists all 17 algorithm variants deployed in the framework. In the *Category* column, *E* denotes Global Exploration, *WS* denotes Cyclic Warm Start, and *G* denotes Greedy Exploitation. The base algorithms used are AILS-II [1], HGS [3], and FILO2 [4].

Table 1: Summary of algorithm variants deployed in the framework.

Method	Category	Description
`e_ails2_1day`	E	AILS-II with 1-day stopping time. Runs once.
`e_ails2_5days`	E	AILS-II with 5-day stopping time. Runs once.
`e_ails2_10days`	E	AILS-II with 10-day stopping time. Runs once.
`e_ails2_20days`	E	AILS-II with 20-day stopping time. Runs once.
`ws_ails2_1day`	WS	AILS-II with 1-day stopping time, cyclic warm start.
`ws_ails2_1.5days`	WS	AILS-II with 1.5-day stopping time, cyclic warm start.
`ws_ails2_2days`	WS	AILS-II with 2-day stopping time, cyclic warm start.
`ws_ails2_3days`	WS	AILS-II with 3-day stopping time, cyclic warm start.
`ws_eoh_ruin`	WS	AILS-II variant (5-day) with EoH-designed ruin operator.
`ws_eoh_omega`	WS	AILS-II variant (5-day) with EoH-designed ω adjustment.
`ws_eta0.01_5days`	WS	AILS-II variant (5-day) with $\eta_{\max} = 0.01$; near-greedy acceptance.
`g_eoh_acc_large`	G	AILS-II variant (1-day) with EoH-designed acceptance criterion (large instances).
`g_eoh_acc`	G	AILS-II variant (1-day) with EoH-designed acceptance criterion.
`g_hgs`	G	HGS [3] with 30-day stopping time.
`g_filo2`	G	FILO2 [4] with 30-day stopping time.
`g_ails2_eta0.02_1day`	G	AILS-II variant (1-day) with $\eta_{\max} = 0.02$.
`g_ails2_eta0.005_1day`	G	AILS-II variant (1-day) with $\eta_{\max} = 0.005$.

The three tiers operate in a complementary temporal pattern across the 30-day competition. In the **early stage** (days 1–5), Greedy Exploitation methods dominate by rapidly converging to competitive solutions and establishing an early leaderboard position. In the **middle stage** (days 5–20), Global Exploration runs with 5- and 10-day stopping times complete and deliver high-quality solutions to the shared database; Cyclic Warm Start methods then intensify the search on top of these solutions, yielding further improvements. In the **late stage** (days 20–30), as the 20-day exploration run matures and earlier jobs free up resources, those resources are reallocated to additional warm-start cycles, shifting the overall search toward deep intensification. This temporal design ensures that each tier contributes most where it has the greatest marginal impact.

3 Computational Resources

All runs were executed on CPU-only servers; no GPU or NPU acceleration was used. Table 2 details the hardware assignments, and Table 3 reports, per instance, the stopping time, number of runs, and total compute time for each method. Each method is single-threaded. The #Runs column in Table 3 reports the total number of independent runs completed on a given instance over the 30 days; when $\#Runs \times stopping\,time$ exceeds 30, some runs were executed in parallel, with the degree of parallelism varying by cluster availability. Aggregated across all 100 competition instances, the total compute time was approximately 42,600 CPU-days (roughly 117 CPU-years). The framework was designed to productively absorb large compute budgets, which was feasible for us given Huawei's internal cluster.

Table 2: Hardware configuration by method group and instance index.

Method Group	Instance Index	CPU	RAM
ws_ails2_1day, ws_ails2_1.5days, ws_ails2_2days, ws_ails2_3days	1–100	Intel Xeon 6972P (2.4 GHz)	Up to 16 GB per run
e_ails2_*, ws_eoh_ruin, ws_eoh_omega, ws_eta0.01_5days, g_eoh_acc_large, g_eoh_acc, g_hgs, g_filo2, g_ails2_eta0.02_1day, g_ails2_eta0.005_1day	1–3	Intel Xeon Gold 6244 (3.60 GHz, 64 cores)	512 GB
	4–9	Intel Xeon Platinum 8280 (2.70 GHz, 112 cores)	512 GB
	10–42	Intel Xeon Platinum 8180M (2.50 GHz, 112 cores)	512 GB
	43–56	Intel Xeon Platinum 8462Y (2.8–4.1 GHz)	512 GB
	57–100	AMD EPYC 9654 (1.5–2.4 GHz, 96 cores)	1536 GB

Table 3: Stopping time, number of runs, and total compute time per method, **reported per instance**. All algorithms are single-core. "#Runs" is the total number of independent runs of the variant completed on a given instance over the 30-day competition. The runs were scheduled partly in parallel (the exact degree of parallelism varied with cluster availability).

Method	Stopping time (days)	#Runs	CPU-days per instance
e_ails2_1day	1	1	1
e_ails2_5days	5	1	5
e_ails2_10days	10	1	10
e_ails2_20days	20	1	20
ws_ails2_1day	1	40	40
ws_ails2_1.5days	1.5	16	24
ws_ails2_2days	2	16	32
ws_ails2_3days	3	8	24
ws_eoh_ruin	5	6	30
ws_eoh_omega	5	6	30
ws_eta0.01_5days	5	6	30
g_eoh_acc_large	1	$\leq$30	$\leq$30
g_eoh_acc	1	$\leq$30	$\leq$30
g_hgs	30	1	30
g_filo2	30	1	30
g_ails2_eta0.02_1day	1	$\leq$30	$\leq$30
g_ails2_eta0.005_1day	1	$\leq$30	$\leq$30
Total	–	–	$\leq$426
Total across 100 instances	–	–	$\leq$ 42,600

4 Competition Outcome

Team OptVerse-CityU won first place in the CVRPLib BKS Challenge with a total score of 1,800.316, holding the best known solution on 51 out of 100 instances at the end of the competition.

The score progression closely followed the temporal design of our framework. Table 4 details the improvement ration of each method in different time window, averaged over 100 instances. In the first two days, the Greedy Exploitation tier—particularly the EoH-designed acceptance criterion variants — rapidly accumulated improvements and established an early lead. Around day 5, the first long-horizon Global Exploration run (e_ails2_5days) completed, and the Cyclic Warm Start tier and Greedy Exploitation tier began intensifying on top of its solutions, producing a second wave of improvements that extended our lead. A similar pattern repeated at days 10 and 20 as the remaining exploration runs matured. Throughout the final week, with all

exploration resources freed and redirected to warm-start cycles, the framework continued to find marginal improvements in the late-stage regime, consolidating the final score.

Table 4: Improvement ratio (%) per variant per time window, averaged over 100 instances.

Method	0–2 days	3–6 days	7–12 days	13–18 days	19–24 days	25–30 days
e_ails2_1day	11.09%	0.00%	0.00%	0.00%	0.00%	0.00%
e_ails2_5days	0.00%	27.18%	0.00%	0.00%	0.00%	0.00%
e_ails2_10days	0.00%	3.10%	20.65%	0.00%	0.00%	0.00%
e_ails2_20days	0.00%	0.00%	1.93%	13.07%	3.61%	0.00%
ws_ails2_1day	0.00%	0.00%	2.19%	44.00%	0.00%	0.00%
ws_ails2_1.5days	0.00%	0.00%	0.00%	17.00%	4.00%	0.00%
ws_ails2_2days	0.00%	0.00%	0.00%	5.00%	13.00%	2.50%
ws_ails2_5days	0.00%	0.00%	0.00%	0.00%	5.00%	4.40%
ws_eoh_ruin	0.00%	2.71%	15.08%	8.70%	32.10%	7.83%
ws_eta0.01_5days	3.90%	1.57%	18.91%	4.64%	8.49%	16.18%
ws_eoh_omega	6.91%	3.50%	0.00%	0.00%	0.00%	0.00%
g_hgs	5.13%	48.28%	38.35%	24.12%	16.12%	10.54%
g_filo2	0.06%	1.38%	3.17%	1.49%	0.35%	0.82%
g_ails2_eta0.02_1day	16.33%	6.34%	1.98%	0.25%	0.00%	0.31%
g_ails2_eta0.005_1day	15.25%	13.60%	7.58%	2.99%	0.13%	0.70%
g_eoh_acc	27.81%	0.00%	0.00%	0.00%	0.00%	0.00%
g_eoh_acc_large	28.67%	0.00%	0.00%	0.00%	0.00%	0.00%

5 Third-Party Components and Acknowledgments

- **AILS-II** [1]: One of the base algorithms (https://github.com/INFORMSJoC/2023.0106)
- **EoH** [2]: LLM-guided automated heuristic design framework (https://github.com/Optima-CityU/llm4ad)
- **HGS** [3]: One of the algorithm (https://github.com/vidalt/HGS-CVRP)
- **FILO2** [4]: One of the base algorithms (https://github.com/acco93/filo2)
- **CVRPLib XLTEST Generator**: Training instances provided by the competition organizers

All novel components—including the three-tier cooperative architecture, EoH-evolved algorithm components, and dynamic resource reallocation strategy—are original contributions of this work.

References

[1] Vinícius R. Máximo, Jean-François Cordeau, and Mariá C. V. Nascimento. AILS-II: An adaptive iterated local search heuristic for the large-scale capacitated vehicle routing problem. *INFORMS Journal on Computing*, 36(4):974–986, 2024.

[2] Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 32201–32223. PMLR, 2024.

[3] Thibaut Vidal. Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood. *Computers & Operations Research*, 140:105643, 2022.

[4] Luca Accorsi and Daniele Vigo. Routing one million customers in a handful of minutes. *Computers & Operations Research*, 164:106562, 2024.