RouteRepair: Instance-Level Failure Diagnosis and Targeted Repair in LLM-Based Automated Heuristic Design for Routing Optimization

Anonymous authors
Paper under double-blind review

Abstract

Large language model (LLM)-based automated heuristic design usually ranks candidate programs according to an aggregate objective, even though a strong routing heuristic may fail repeatedly on a coherent subset of instances. We introduce RouteRepair, a fixed-backbone framework that retains the instance-wise performance profile of each competitive parent, constructs parent-specific failure, strength, and protection sets, and uses this evidence to define a bounded program repair. A Failure Diagnosis Expert produces a structured, risk-aware repair brief; a separate Targeted Repair Expert modifies only the designated scoring rule or edge-prior function. Parent and child programs are evaluated on matched instances, solver seeds, settings, and budgets, and both successful and unsuccessful interventions are recorded in outcome-aware memory. We evaluate five adapters covering constructive TSP, guided local search (GLS) for TSP, constructive CVRP, ant colony optimization (ACO) for TSP, and ACO for CVRP. RouteRepair-GLS reduces the mean TSP gap from 1.7476% to 0.7587%. Under identical 2-opt postprocessing, the constructive TSP rule achieves a 2.8805% mean gap. Constructive CVRP lowers average route cost by 1.91% relative to the Savings heuristic and by 2.06% relative to the strongest LLM-evolved baseline. The ACO priors also improve matched hand-designed priors and retain gains on larger transfer instances. These results support instance-level, evidence-constrained repair as a practical complement to aggregate evolutionary search.

1 Introduction

Routing optimization supports freight distribution, urban logistics, shared mobility, and autonomous delivery. Exact algorithms remain essential reference tools, but constructive heuristics, local search, and population-based metaheuristics are often preferred when high-quality routes are required under limited computational budgets (Laporte, 2009; Toth & Vigo, 2014; Vidal et al., 2014). Their effectiveness can depend on compact components: a constructor ranks feasible customers, guided local search (GLS) chooses route features to penalize, and ant colony optimization (ACO) uses an edge prior to guide stochastic construction.

Automated heuristic design (AHD) searches over such rules rather than directly solving one instance. Code-capable LLMs broaden the program prior and support semantic variation, as demonstrated by FunSearch, EoH, and ReEvo (Romera-Paredes et al., 2024; Liu et al., 2024; Ye et al., 2024). Yet most LLM-based AHD pipelines still compress performance into a mean objective or average gap. This scalar cannot distinguish a uniformly mediocre rule from a competitive rule that repeatedly fails on a small but structurally coherent region of the instance space.

RouteRepair addresses this limitation by retaining the full instance-wise outcome profile of each selected parent. The method identifies parent-specific failure cases, extracts strength cases that summarize behavior worth preserving, and reserves the remaining non-failure

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

instances as a protection set. A diagnosis expert links observed route behavior and instance descriptors to a bounded repair objective. A separate repair expert then edits only the declared heuristic interface. Matched parent–child evaluation determines whether the intervention recovers failure cases without unacceptable collateral degradation.

Our contributions are threefold. First, we formulate heuristic refinement as an evidence-constrained intervention rather than aggregate candidate replacement. Second, we introduce a two-expert diagnosis–repair protocol with frozen repair briefs, protection-set validation, and outcome-aware memory. Third, we demonstrate the same mechanism across five TSP and CVRP adapters with fixed constructive, GLS, and ACO backbones. We report scale-wise, transfer, ablation, and instance-level results, including cases where gains are modest or non-universal.

2 Related Work

Routing and learned components. TSP and CVRP have long served as testbeds for constructive heuristics, guided local search, ant-colony methods, and hybrid metaheuristics (Applegate et al., 2006; Voudouris & Tsang, 1999; Dorigo & Gambardella, 1997; Laporte, 2009; Toth & Vigo, 2014; Vidal et al., 2014). The surrounding solver determines the broad search process, while compact scoring and selection rules determine how effectively it uses the available budget. Learning-assisted routing usually preserves an explicit optimization structure and adapts one component of the decision process. Examples include graph-guided local search, joint matching, routing, and pricing for taxi sharing, real-time on-demand bus routing, reinforcement learning for truck–drone delivery, hierarchical dispatching, and learning-guided decomposition for large vehicle-routing instances (Hudson et al., 2022; Qiu et al., 2022; Lian et al., 2023; Bogyrbayeva et al., 2023; Si et al., 2024; Kerscher & Minner, 2025; Li et al., 2025). RouteRepair follows the same fixed-backbone principle, but its learned object is executable heuristic code rather than an end-to-end routing policy.

LLM-based automated heuristic design. Hyper-heuristics and genetic programming established the search-over-heuristics perspective, although their search spaces commonly rely on manually defined primitives (Burke et al., 2013; Qu et al., 2020; Dokeroglu et al., 2024). Evolution through Large Models, Language Model Crossover, and EvoPrompting showed that pretrained models can act as semantic variation operators over programs (Lehman et al., 2024; Meyerson et al., 2024; Chen et al., 2023). FunSearch couples program sampling with evaluator-guided selection; EoH jointly evolves heuristic ideas and code; and ReEvo uses comparative reflection as a verbal search signal (Romera-Paredes et al., 2024; Liu et al., 2024; Ye et al., 2024). Benchmarking further shows that the organization of evolutionary search materially affects the quality of generated heuristics (Zhang et al., 2024).

Subsequent methods broaden the design target and reuse information more explicitly. Planning- and MCTS-based methods allocate evaluations toward promising branches, while knowledge-guided prompting carries information across candidate generations (Wang et al., 2025; Zheng et al., 2025; Wu et al., 2025). EoH-S searches for complementary heuristic sets; RedAHD reduces dependence on a manually chosen algorithmic framework; AlphaEvolve evolves larger code regions; HeurAgenix combines generation with state-dependent selection (Liu et al., 2025; Thach et al., 2025; Novikov et al., 2025; Yang et al., 2025); and meta-optimization is used to improve the generalizability of generated heuristics (Shi et al., 2026). Structured failure feedback and repair prompts have also been applied to vehicle–drone collaborative routing (Shi & Zhen, 2026). RouteRepair is complementary: it starts from a competitive parent, diagnoses its recurrent instance-level weakness, and tests a localized intervention without allowing the generated program to replace the fixed routing backbone.

Instance heterogeneity and repair. Instance-space analysis shows that algorithm performance depends strongly on structural properties (Smith-Miles & Muñoz, 2023). Recent evaluation of LLM-evolved heuristics reaches a related conclusion: a strong average may reflect specialization rather than broad superiority (Sim et al., 2025). Portfolios and dynamic selectors address heterogeneity by deploying different rules in different regions, whereas RouteRepair revises one heuristic while preserving its established strengths. The approach

is related to verbal-reflection agents and automated program repair (Shinn et al., 2023; Madaan et al., 2023; Le Goues et al., 2012; Monperrus, 2018; Olausson et al., 2024), but routing rarely supplies a binary pass/fail signal. A program may remain feasible yet produce poor solutions only on part of the distribution. The repair target must therefore be defined from continuous solution-quality evidence, and collateral effects must be checked on instances the parent already solves well. Matched parent–child execution, rather than the LLM's explanation, determines whether an intervention is a genuine repair.

3 RouteRepair

3.1 Problem Setting

Let $\mathcal{D} = \{x_i\}_{i=1}^{N}$ be a set of routing instances. Adapter k provides a fixed solver S_k and objective f_k; the generated heuristic h changes only a declared scoring or edge-prior function. The resulting solution and objective are

$$y_i(h) = S_k(x_i; h), \qquad z_i(h) = f_k(y_i(h)). \tag{1}$$

All objectives are minimized. For reference value z_i^{ref}, the normalized excess cost is

$$r_i(h) = \frac{z_i(h) - z_i^{\mathrm{ref}}}{\max(|z_i^{\mathrm{ref}}|, \epsilon)}, \qquad \epsilon > 0. \tag{2}$$

RouteRepair retains both $\mathbf{r}(h) = (r_1(h), \ldots, r_N(h))$ and $\bar{r}(h) = N^{-1} \sum_i r_i(h)$. The scalar supports outer-loop ranking; the vector supports localized diagnosis.

3.2 Nested Search Framework

Figure 1 summarizes the method. The outer loop generates executable candidates, rejects invalid programs, evaluates valid candidates within a fixed routing backbone, and selects competitive parents. The inner loop mines the selected parent's failures, diagnoses the recurrent weakness, creates bounded repairs, and returns validated children to the population. Outcome-aware memory records the diagnosis, edit, measured recovery, collateral degradation, and validation status. Failed repairs remain useful negative evidence instead of being converted into positive instructions.

3.3 Parent-Specific Evidence Sets

Within each problem scale g, parent outcomes are ordered by severity. If m_g is the number of instances at scale g and rank 1 is worst, the normalized severity rank is

$$s_i(h_p) = \frac{m_{g_i} - \mathrm{rank}_{g_i}(i; h_p)}{\max(m_{g_i} - 1, 1)}. \tag{3}$$

The failure set $\mathcal{F}(h_p)$ contains the highest-severity fraction within each scale. A compact strength set $\mathcal{S}(h_p)$ is sampled from the best-performing tail and is shown to the diagnosis expert. The full protection set $\mathcal{A}(h_p) = \mathcal{D} \setminus \mathcal{F}(h_p)$ is withheld for the external collateral-damage test. When a matched baseline exists, contrastive losses enrich the evidence but do not replace parent-specific tail selection.

Figure 2 illustrates why the three sets play different roles. Failure cases define the intervention target; strength cases summarize behavior worth preserving; the broader protection set prevents a small displayed sample from masking newly created failures.

3.4 Diagnosis, Repair, and Verification

The Failure Diagnosis Expert receives the parent code, failure and strength summaries, route-quality records, instance descriptors, solver behavior, and the exact adapter contract. It produces a structured report containing failure evidence, weak behavior, a testable root-cause hypothesis, repair objective, preservation risk, and one to three distinct repair families.

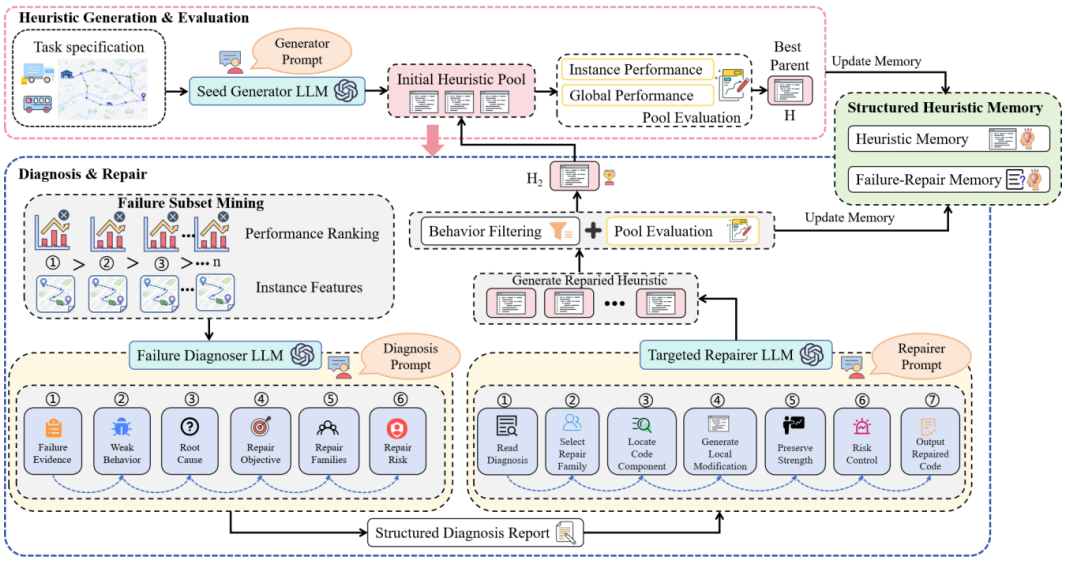


Figure 1: RouteRepair combines outer heuristic generation and evaluation with an inner diagnosis–repair loop. Only the adapter-specific scoring rule or edge-prior function is editable; feasibility logic and solver dynamics remain fixed.

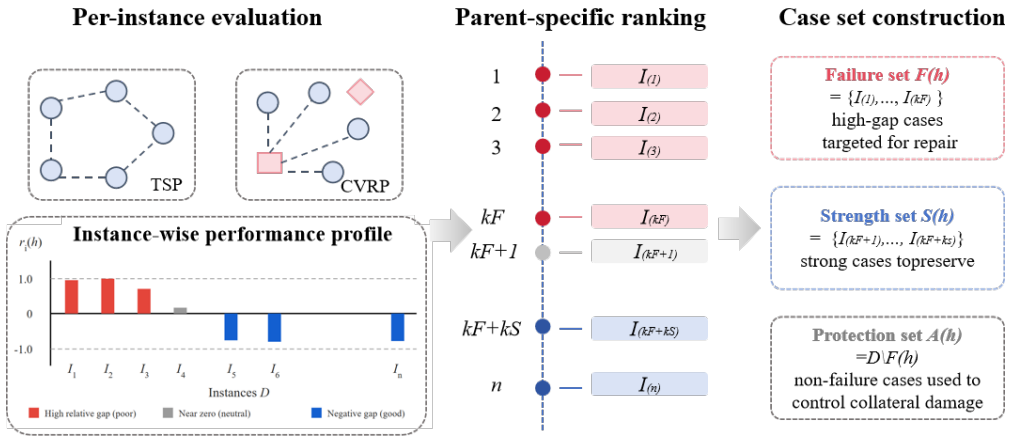


Figure 2: Instance-wise outcomes are ranked within scale to create parent-specific failure and strength evidence. All non-failure instances form the protection set used for matched validation.

An internal risk review removes unsupported or duplicate directions and freezes a concise brief before code generation.

The Targeted Repair Expert receives the frozen brief, one selected repair family, the parent code, compact evidence, elite context, and relevant successful and failed memory records. It must preserve the function signature and remain inside the fixed-backbone boundary. Constructive and GLS adapters return one finite score; ACO adapters return a deterministic finite matrix with the same shape as the distance matrix. Pheromone dynamics, route construction, feasibility checks, and unavailable state cannot be modified.

After syntax, interface, shape, and numerical validation, child h_c is compared with parent h_p under matched conditions. Define failure gain, normalized recovery, and one-sided collateral

degradation as

$$G_F = \frac{1}{|\mathcal{F}|} \sum_{i \in \mathcal{F}} [r_i(h_p) - r_i(h_c)], \tag{4}$$

$$R_F = \frac{G_F}{\frac{1}{|\mathcal{F}|} \sum_{i \in \mathcal{F}} r_i(h_p) + \epsilon}, \tag{5}$$

$$C_A = \frac{1}{|\mathcal{A}|} \sum_{i \in \mathcal{A}} \max\{0, r_i(h_c) - r_i(h_p)\}. \tag{6}$$

A child is a verified targeted repair only if $R_F \geq \tau_F$ and $C_A \leq \tau_A$. Population entry remains separate: a child with a useful aggregate trade-off may survive search without being counted as a verified repair.

4 Experimental Evaluation

4.1 Setup

We evaluate five adapters: TSP-GLS, constructive TSP, constructive CVRP, TSP-ACO, and CVRP-ACO. Experiments use Python 3.10.19 and the API configuration identified as deepseek-v4-flash. The same model performs initial generation, diagnosis, risk review, and repair; no task-specific parameter training is used. Constructive TSP and CVRP searches use 128 training instances, 25 evolutionary rounds, eight initial candidates, and up to five repairs per round. Every parent–child comparison fixes instances, solver seeds, configurations, and evaluation budgets. Full adapter interfaces, data splits, reference policies, and ACO budgets appear in Appendix C.

We compare only under declared interfaces. "Lite" or "adapted" labels denote component-level transfers into the common backbone, not reproductions of full source systems. OR-Tools Routing GLS is an external solver reference for constructive CVRP because it adds neighborhood search beyond the matched next-customer interface. Reported transfer pools retain all available instances, including those on which RouteRepair is not best.

4.2 TSP Results

The scaffold comparison includes component-level adaptations of EB-GLS (Shi et al., 2018) and GNN-guided local search (Hudson et al., 2022) into the common GLS backbone. Under the common GLS scaffold, RouteRepair-GLS achieves a 0.7587% average gap, compared with 1.7476% for standard GLS and 0.9079% for the best initial LLM rule. The largest scale-wise gain is on TSP200, where the gap falls from 3.5791% to 1.6897%. In the complete procedure comparison, RouteRepair reaches a 0.6587% average gap over TSP20–TSP200 with 216 LLM calls, whereas the tested ReEvo campaign reaches 0.6861% with 302 calls. This comparison is specific to the tested implementations and budgets. Under the scaffold configuration, RouteRepair-GLS also reaches a 2.3516% mean gap on the held-out TSP500 generalization scale (Table 5), down from 4.8924% for standard GLS and comparable to the 2.3439% of the full ReEvo campaign.

For constructive TSP, RouteRepair's pure constructor is not uniformly superior to farthest insertion. The benefit becomes clear after applying the same 2-opt procedure to every starting tour: RouteRepair obtains the lowest mean gap, 2.8805%, and the best result at each evaluated scale. On the complete 30-instance TSPLIB transfer pool, RouteRepair-GLS achieves a 1.8742% mean gap, while RouteRepair construction plus 2-opt obtains 3.9244%.

The TSP-ACO adapter changes only a static edge-desirability matrix. Across TSP20–TSP200, its equal-scale average gap is 5.145%, compared with 5.341% for the ReEvo prior under the common ACO backbone. Neither dominates every scale: RouteRepair is better on TSP50 and TSP100, while ReEvo is slightly better on TSP20, TSP200, and TSP500.

Table 1: Selected aggregate results. Lower is better. Values reproduce the matched evaluations; task metrics differ and should not be compared across rows.

Adapter	Metric	RouteRepair	Matched comparator
TSP-GLS	mean gap (%)	0.7587	1.7476 (standard GLS)
Constructive TSP + 2-opt	mean gap (%)	2.8805	3.3430 (NN + 2-opt)
Constructive CVRP	mean route cost	26,723.6	27,243.6 (savings)
TSP-ACO	TSP20–200 equal-scale gap (%)	5.145	5.341 (ReEvo)
CVRP-ACO	CVRP50–500 mean cost	28,184.9	29,253.0 (capacity-aware)

4.3 CVRP Results

Within the capacity-feasible constructive interface, RouteRepair obtains the lowest held-out average route cost, 26,723.6. This is 1.91% below savings, 2.06% below the ReEvo-adapted guide, and 7.75% below the best initial LLM program. On the complete CVRPLIB pool, the gap to best-known solutions is 23.001%, compared with 23.144% for the ReEvo-adapted guide and 24.835% for savings; the small difference from ReEvo is interpreted as preserved transfer competitiveness rather than a large gain.

For CVRP-ACO, RouteRepair lowers mean cost at every held-out scale. Across 16 CVRP50–CVRP500 instances, it wins 14 comparisons and reduces mean cost from 29,253.0 to 28,184.9, a 3.32% reduction. It also wins all four CVRP1000 transfer comparisons, reducing mean cost from 103,785.8 to 100,039.1. Vehicle counts and mean utilization are unchanged, so the gain arises from edge ordering within the same feasible construction process.

4.4 Repair Behavior

Ablations on TSP-GLS yield average gaps of 0.7587% for full RouteRepair, 0.9285% without expert-role separation, 1.7040% without memory, 1.7385% without repair, and 1.7469% without failure attribution. Differences are small on TSP20 and grow on TSP100 and TSP200, suggesting that the evidence-to-repair loop matters more as scale increases. The ablation uses one controlled seed and therefore supports a mechanism-oriented comparison rather than a claim about variance across independent campaigns.

Figure 3 shows that the aggregate gains are not produced by only a few favorable cases. With a tolerance of 0.1 percentage points, RouteRepair improves 158/256 TSP-GLS instances, 253/256 constructive CVRP instances, 13/16 TSP-ACO instances, and 14/16 CVRP-ACO instances. Degraded subsets remain, particularly for the ACO adapters, and are explicitly retained in the analysis. Full numerical tables and the remaining result figures are provided in Appendix D.

Figure 4 shows the best-so-far and population-median training gaps over the 25 evolutionary rounds for the TSP-GLS and constructive TSP adapters. Both curves decrease within the budgeted rounds, and the gap between the population median and the best-so-far reflects the selective pressure of the outer loop.

5 Limitations

RouteRepair does not dominate every method, scale, or instance. Several comparisons are modest and some are component-level adaptations rather than full-system reproductions. The principal ablation is based on a single controlled seed; repeated-seed uncertainty is not available for every adapter. Reference values also differ by protocol: exact or best-known values are used where available, whereas some large-scale comparisons use fixed baseline-derived references. The generated component is deliberately narrow, so improvements do not imply superiority over solvers that add stronger neighborhoods, learned policies, or larger search budgets. Finally, the diagnosis is an intervention hypothesis, not a causal explanation; only matched execution validates the proposed repair.

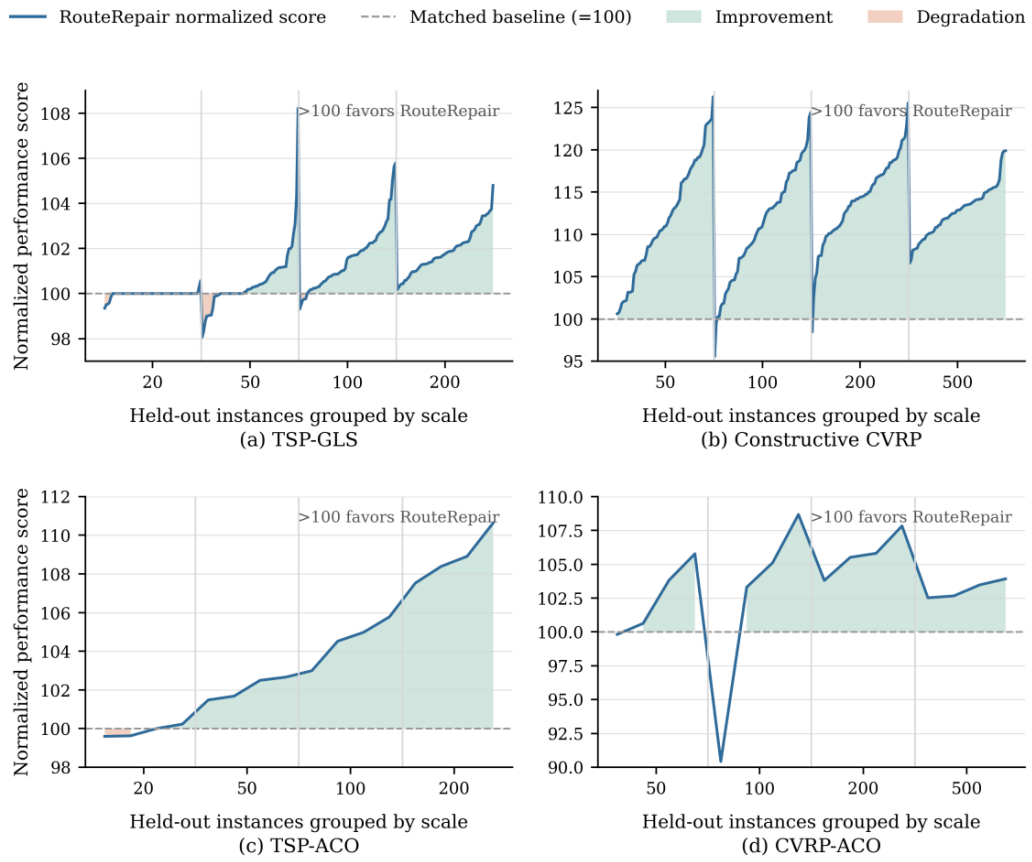


Figure 3: Instance-wise normalized performance profiles for the four adapters: (a) TSP-GLS, (b) constructive CVRP, (c) TSP-ACO, and (d) CVRP-ACO. Each matched baseline is normalized to 100; values above 100 favor RouteRepair and values below 100 indicate degradation.

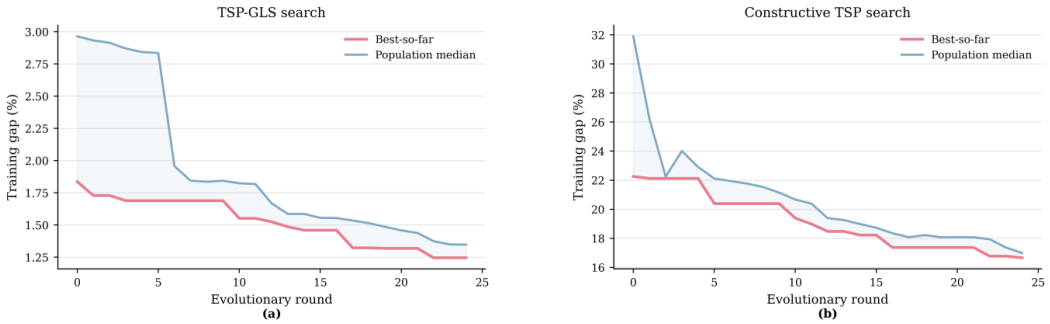


Figure 4: Best-so-far and population-median training gaps across the 25 evolutionary rounds for the TSP-GLS and constructive TSP adapters.

6 Conclusion

RouteRepair changes LLM-based heuristic refinement from aggregate replacement to instance-level, evidence-constrained repair. It preserves a parent's full outcome profile, diagnoses recurrent weaknesses, applies bounded edits through a separate repair expert, and verifies both recovery and collateral damage under matched conditions. Across five TSP and

CVRP adapters, this mechanism improves competitive constructive, GLS, and ACO components while retaining negative and scale-dependent outcomes. The broader implication is that automated heuristic design benefits from treating local failures as testable intervention targets rather than information erased by a mean score.

Reproducibility Statement

Appendix C specifies the five editable interfaces, fixed solver boundaries, data splits, references, search budgets, and validation rules. Appendix E records the diagnosis and repair contracts, and Appendix D provides the full quantitative tables and additional figures. An anonymized code archive accompanies the submission and contains the fixed backbones, shared diagnosis–repair loop, prompt templates, and smoke tests; historical outputs, credentials, and author metadata are excluded.

AI Use Statement

Generative AI tools were used in this work in two roles. The LLM backend of the studied experiments (the deepseek-v4-flash API, Section 4.1) served as the program-generating and program-repair agent, and the same class of tools also assisted with language revision and document formatting. They were not used to replace executed evaluations or to invent numerical results, citations, or experimental records. All reported measurements come from the authors' executed experiment records, and AI-assisted content was reviewed by the authors, who take responsibility for the final paper.

References

David L. Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook. The Traveling Salesman Problem: A Computational Study. Princeton University Press, 2006.

Aigerim Bogyrbayeva, Taehoon Yoon, Hyeonjoon Ko, Sanghoon Lim, Hyunseok Yun, and Changhyun Kwon. A deep reinforcement learning approach for solving the traveling salesman problem with drone. Transportation Research Part C: Emerging Technologies, 148:103981, 2023. doi: 10.1016/j.trc.2022.103981.

Edmund K. Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: A survey of the state of the art. Journal of the Operational Research Society, 64(12):1695–1724, 2013.

Angelica Chen, David M. Dohan, and David R. So. Evoprompting: Language models for code-level neural architecture search. In Advances in Neural Information Processing Systems, volume 36, 2023.

Tansel Dokeroglu, Tayfun Kucukyilmaz, and El-Ghazali Talbi. Hyper-heuristics: A survey and taxonomy. Computers & Industrial Engineering, 187:109815, 2024.

Marco Dorigo and Luca Maria Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1(1):53–66, 1997.

Benjamin Hudson, Qingbiao Li, Matthew Malencia, and Amanda Prorok. Graph neural network guided local search for the traveling salesperson problem. In International Conference on Learning Representations, 2022.

Christian Kerscher and Stefan Minner. Decompose-route-improve framework for solving large-scale vehicle routing problems with time windows. Transportation Research Part E: Logistics and Transportation Review, 204:104409, 2025.

Gilbert Laporte. Fifty years of vehicle routing. Transportation Science, 43(4):408–416, 2009. doi: 10.1287/trsc.1090.0301.

Claire Le Goues, ThanhVu Nguyen, Stephanie Forrest, and Westley Weimer. Genprog: A generic method for automatic software repair. IEEE Transactions on Software Engineering, 38(1):54–72, 2012.

Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O. Stanley. Evolution through large models. In Handbook of Evolutionary Machine Learning. Springer, 2024.

S. Li, E. I. Kaisar, D. Kwak, B. Hu, and D. Liu. Optimizing heterogeneous capacitated vehicle routing with linformer and multi-relationship decoding: A deep reinforcement learning approach. Transportation Research Record, 2679(10):39–60, 2025. doi: 10.1177/03611981251338719.

Yujie Lian, Florian Lucas, and Kenneth Sørensen. The on-demand bus routing problem with real-time traffic information. Multimodal Transportation, 2(3):100093, 2023.

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhaojie 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, 2024.

Fei Liu, Y. Liu, Qingfu Zhang, Xialiang Tong, and Mingxuan Yuan. Eoh-s: Evolution of heuristic set using llms for automated heuristic design. arXiv preprint arXiv:2508.03082, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, et al. Self-refine: Iterative refinement with self-feedback. In Advances in Neural Information Processing Systems, volume 36, 2023.

Elliot Meyerson, Mark J. Nelson, Herbie Bradley, Adam Gaier, Arash Moradi, Amy K. Hoover, and Joel Lehman. Language model crossover: Variation through few-shot prompting. ACM Transactions on Evolutionary Learning and Optimization, 4(4):Article 27, 2024.

Martin Monperrus. Automatic software repair: A bibliography. ACM Computing Surveys, 51(1):1–24, 2018.

Alexander Novikov, Nguyễn Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Z. Wagner, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. arXiv preprint arXiv:2506.13131, 2025.

Theo X. Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. Is self-repair a silver bullet for code generation? In Proceedings of the International Conference on Learning Representations, 2024.

Jing Qiu, Kai Huang, and Jason Hawkins. The taxi sharing practices: Matching, routing and pricing methods. Multimodal Transportation, 1(1):100003, 2022.

Rong Qu, Graham Kendall, and Nelishia Pillay. The general combinatorial optimization problem: Towards automated algorithm design. IEEE Computational Intelligence Magazine, 15(2):14–23, 2020.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, et al. Mathematical discoveries from program search with large language models. Nature, 625:468–475, 2024.

Hao Shi and Lu Zhen. Llm-based automatic heuristic design for vehicle-drone collaborative routing problems. Transportation Research Part E: Logistics and Transportation Review, 209:104760, 2026.

Jialong Shi, Qingfu Zhang, and Edward Tsang. Eb-gls: An improved guided local search based on the big valley structure. Memetic Computing, 10(3):333–350, 2018.

Yuhui Shi, Jianan Zhou, Wen Song, Jie Bi, Yue Wu, Zhiguang Cao, and Jie Zhang. Generalizable heuristic generation through llms with meta-optimization. In Proceedings of the International Conference on Learning Representations, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In Advances in Neural Information Processing Systems, volume 36, 2023.

Jia Si, Fei He, Xiaohong Lin, and Xiaowo Tang. Vehicle dispatching and routing of on-demand intercity ride-pooling services: A multi-agent hierarchical reinforcement learning approach. Transportation Research Part E: Logistics and Transportation Review, 186: 103551, 2024.

Kevin Sim, Quentin Renau, and Emma Hart. Beyond the hype: Benchmarking llm-evolved heuristics for bin packing. In Applications of Evolutionary Computation, pp. 386–402, 2025.

Kate Smith-Miles and Mario A. Muñoz. Instance space analysis for algorithm testing: Methodology and software tools. ACM Computing Surveys, 55(12):1–36, 2023.

Nguyen Thach, Ali Riahifar, Nhat Huynh, and Huy Chan. Redahd: Reduction-based end-to-end automatic heuristic design with large language models. arXiv preprint arXiv:2505.20242, 2025.

Paolo Toth and Daniele Vigo (eds.). Vehicle Routing: Problems, Methods, and Applications. Society for Industrial and Applied Mathematics, 2 edition, 2014.

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. A unified solution framework for multi-attribute vehicle routing problems. European Journal of Operational Research, 234(3):658–673, 2014.

Chris Voudouris and Edward Tsang. Guided local search and its application to the travelling salesman problem. European Journal of Operational Research, 113(2):469–499, 1999.

Hao Wang, Xin Zhang, and Chunyang Mu. Planning of heuristics: Strategic planning on large language models with monte carlo tree search for automating heuristic optimization. arXiv preprint arXiv:2502.11422, 2025.

Xinyue Wu, Dong Wang, Chenyang Wu, Lijun Wen, Chunyan Miao, Yang Xiao, and Yu Zhou. Efficient heuristics generation for solving combinatorial optimization problems using large language models. In Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2025.

X. Yang, L. Zhang, H. Qian, L. Song, and J. Bian. Heuragenix: Leveraging llms for solving complex combinatorial optimization challenges. arXiv preprint arXiv:2506.15196, 2025.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. Reevo: Large language models as hyper-heuristics with reflective evolution. In Advances in Neural Information Processing Systems, volume 37, 2024.

Rui Zhang, Fei Liu, Xi Lin, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Understanding the importance of evolutionary search in automated heuristic design with large language models. In Parallel Problem Solving from Nature – PPSN XVIII, 2024.

Zeyu Zheng, Zhenyu Xie, Zhenkun Wang, and Bryan Hooi. Monte carlo tree search for comprehensive exploration in llm-based automatic heuristic design. In Proceedings of the International Conference on Machine Learning, 2025.

Table 2: Fixed routing backbones and the only LLM-generated component in each adapter.

Adapter	Fixed backbone	Generated function
TSP-GLS	Guided local search	score_edge_for_penalty(city_a, city_b, path, distance_matrix, penalties, coords, state)
Constructive TSP	Deterministic constructor	score_next_node(current_node, candidate_node, unvisited_nodes, distance_matrix, tour_history)
Constructive CVRP	Capacity-feasible constructor	score_next_customer(current_node, candidate_customer, unserved_customers, remaining_capacity, distance_matrix, demands, depot)
TSP-ACO	Fixed ACO	score_tsp_aco_edges(distance_matrix)
CVRP-ACO	Fixed feasible ACO	score_cvrp_aco_edges(distance_matrix, coordinates, demands, vehicle_capacity, depot)

A Adapter Contracts and Diagnostic Evidence

The fixed solver controls feasibility, candidate construction, state transitions, neighborhood operations, pheromone updates, random seeds, stopping criteria, and evaluation budgets. The LLM cannot add a hidden local search or use state absent from the declared signature. TSP evidence includes size, distance distribution, nearest-neighbor spacing, local/global spacing ratio, density, aspect ratio, separation, long-edge ratio, crossings, stagnation, and score/prior distributions. CVRP additionally includes depot-radial dispersion, demand variation, demand/capacity pressure, spatial-demand correlation, route/load variance, fragmentation, utilization, and late-customer patterns.

For the ACO adapters, the generated matrix $H_h = h(D; \xi) \in \mathbb{R}^{n \times n}$ is validated, converted to a finite nonnegative desirability matrix η_h, and combined with pheromone τ by the fixed transition rule

$$P(j \mid i, t) \propto [\tau_{ij}^{(t)}]^{\alpha} [\eta_{h,ij}]^{\beta}, \qquad j \in \mathcal{U}_t. \tag{7}$$

The generated prior cannot change α, β, evaporation, deposits, ant counts, iterations, or capacity feasibility.

B Evidence Construction and Repair Acceptance

B.1 Scale-balanced case mining

Failure and strength cases are selected within each problem scale rather than from one global ranking. This prevents larger or intrinsically harder instances from occupying the entire diagnosis context. The failure pool contains the highest-severity 20% within each scale, subject to a minimum of three cases. The displayed strength set is sampled from the best-performing tail. The protection set is larger: it contains every non-failure case and is evaluated externally even though only a compact strength summary is shown to the language model. Thus, strength cases are prompt evidence, whereas protection cases implement the acceptance constraint.

For TSP-GLS, severity is the normalized within-scale gap rank augmented by the matched Standard-GLS comparison. Constructive TSP ranks primarily by optimality gap and then by longest-edge ratio. Constructive CVRP uses gap together with vehicle count, load variance, and low utilization. ACO evidence includes the same instance descriptors plus prior-distribution and route-construction summaries. These secondary fields diagnose behavior; the executed objective remains the source of the reported result.

B.2 Diagnosis and repair-family separation

The diagnosis stage receives parent code, failure and strength records, the top population context, and the immutable adapter contract. Its structured output distinguishes observed evidence, weak behavior, a root-cause hypothesis, a repair objective, preservation risks, and

candidate repair families. A separate risk review removes unsupported directions before the repair expert sees the brief. This separation is intended to prevent the code generator from silently redefining the failure after seeing a candidate implementation.

Up to five repair roles are used: a conservative targeted patch, hybrid recombination with an elite mechanism, a structural scoring rewrite, a counterfactual hypothesis test, and a bounded stagnation-escape mutation. These roles diversify the intervention while preserving the same function signature and fixed solver. Recent memory is bounded to at most six repair records and includes both successful and failed interventions, their measured failure gain, collateral change, validation status, and rejection reason.

B.3 Static and matched-execution validation

Before full evaluation, a candidate must expose the exact function name and ordered arguments, contain exactly one generated function, avoid imports and unsafe calls, return the required scalar or matrix shape, produce finite values, avoid input mutation, and pass a task-level smoke test. Constructive candidates are additionally checked for prohibited full-set scans or hidden route optimization. These checks establish interface validity, not solution quality.

Valid children are then executed on the same instances, seeds, solver settings, and budgets as their parent. A repair is marked successful only when its normalized failure recovery is at least 0.05 and its average one-sided degradation on the protection set is no more than 0.02 absolute gap. Constructive TSP uses the archived 0.03 protection trade-off limit. These thresholds determine the repair label; outer-loop survivor selection remains a separate decision.

C Experimental Protocol

C.1 Search and evaluation budgets

Parent and child use identical instances, seeds, settings, and budgets. Constructive TSP and CVRP searches use 128 training instances, 25 rounds, eight initial candidates, and at most five repair candidates per round. The archived language-model configuration records temperature 0.7, up to three request attempts, and a 60-second request timeout. Model output never determines acceptance: every reported comparison comes from external execution.

TSP-ACO uses 24 ants, 40 iterations, and solver seeds 0, 1, and 2. Held-out TSP-ACO evaluation contains four instances each at TSP20, TSP50, TSP100, and TSP200, plus two TSP500 transfer instances. Constructive CVRP contains 64 held-out instances at each of CVRP50, CVRP100, CVRP200, and CVRP500. CVRP-ACO reports four instances per scale for CVRP50–CVRP500 and four CVRP1000 transfer instances. All scale averages are first computed within the declared split; values from different task metrics are not pooled.

C.2 Split and reference discipline

Random TSP training data contain 32 instances each at TSP20, TSP50, TSP100, and TSP200; held-out evaluation contains 64 instances at each scale, and TSP500 is reserved for transfer. Constructive CVRP uses the same four training scales with a balanced 64-instance evolution subset, 64 held-out instances per scale, and CVRP1000 for transfer. Public TSPLIB and CVRPLIB evaluations retain the complete available pools used by the archived protocol rather than selecting only favorable instances.

TSP gaps use exact or established optimal values under the corresponding protocol. TSPLIB and CVRPLIB transfer pools retain all available instances. Constructive comparisons apply an identical 2-opt implementation when postprocessing is reported. ACO comparisons hold the ACO backbone fixed and change only the prior. External solvers that introduce additional neighborhoods are reported as references rather than matched-interface baselines.

For a trusted optimum or best-known value f_i^{ref}, minimization gap is $100(f_i - f_i^{\text{ref}})/f_i^{\text{ref}}$. When a stored reference is produced by another heuristic rather than an optimum, the paper reports improvement relative to that reference and does not call it an optimality gap. Runtime is reported for executed solvers; language-model latency and call counts are tracked separately because they measure search cost rather than route-construction time.

C.3 Reproducibility records

Each archived campaign records a run manifest and framework contract, model call metadata, executable candidate source, candidate registry, diagnosis and reflection outputs, failure-repair memory, aggregate and instance-level evaluation files, round logs, and dataset manifests. Dataset manifests bind instance identifiers to split, scale, seed, file hash, reference type, reference method, and reference value. These records allow the reported aggregate tables to be traced back to matched instance-level evaluations while keeping credentials and cached service responses outside the submission package.

D Full Quantitative Results

This section reports task-specific metrics without combining them across adapters. Bold values identify the best value within the displayed matched comparison, not universal solver dominance. Differences between a fixed-component comparison and a complete external solver are stated explicitly.

Table 3: TSP-GLS under a common solver scaffold (optimality gap, %; lower is better).

Method	Avg.	TSP20	TSP50	TSP100	TSP200
Standard GLS	1.7476	0.0150	0.8693	2.5271	3.5791
Low-penalty GLS	1.8323	0.0690	0.9966	2.5827	3.6807
EBGLS-lite	1.7578	0.0614	1.0533	2.3650	3.5516
GNNGLS-lite	1.7554	0.0407	1.0506	2.4018	3.5285
EoH	2.2077	0.2713	1.7217	3.0113	3.8266
PoH	1.7066	0.0444	0.9466	2.3557	3.4797
RouteRepair-GLS	0.7587	0.0358	0.3343	0.9751	1.6897

Interpretation. RouteRepair-GLS attains the best average gap (0.7587%) and the best TSP50, TSP100, and TSP200 values, while its TSP20 gap (0.0358%) remains close to standard GLS (0.0150%). The relative gain over standard GLS grows from TSP50 (0.3343% vs 0.8693%) to TSP200 (1.6897% vs 3.5791%), consistent with failure-driven repair becoming more valuable as instances become harder.

Table 4: Budget-aware comparison with the tested ReEvo campaign.

Method	LLM calls	Avg. gap	TSP200	TSP500
Standard GLS	0	1.7476	3.5791	4.8924
ReEvo	302	0.6861	1.9726	2.3439
RouteRepair	216	0.6587	1.6897	2.2916

Interpretation. RouteRepair uses 216 LLM calls, 86 fewer than the tested ReEvo campaign (302), while achieving the lowest average gap (0.6587%) and the best TSP200 and TSP500 values. This comparison is specific to the tested campaign configurations.

TSP500 is reserved as a held-out generalization scale (64 instances, none used in training or in the scaffold comparison). The table below reports this generalization for the scaffold run; the complete-procedure numbers in Table 4 come from a different run with a different call budget and should not be combined row-wise.

Interpretation. On the held-out TSP500 scale, RouteRepair-GLS (2.3516%) and the full ReEvo campaign (2.3439%) are nearly identical, and both are well below the EoH pilot

Table 5: TSP500 generalization under the scaffold run (64 transfer instances; gap, %; lower is better).

Method	Avg. gap	Median gap	Avg. runtime (s)
RouteRepair-GLS (176 LLM calls)	2.3516	2.3175	—
ReEvo official full (302 calls)	2.3439	2.2899	3.4154
EoH official pilot	2.6654	2.7551	24.9045
Standard GLS	4.8924	4.7341	—

(2.6654%) and standard GLS (4.8924%). The runtime column reports executed solver time only; LLM latency is excluded.

Table 6: Constructive TSP before and after identical 2-opt (optimality gap, %).

Method	Avg.	TSP20	TSP50	TSP100	TSP200
Nearest Neighbor	22.5292	16.6312	23.3894	24.0871	26.0093
Cheapest Insertion	16.2027	10.5209	16.4571	18.2242	19.6084
Farthest Insertion	6.0049	2.1389	5.3997	7.2928	9.1883
NN + 2-opt	3.3430	1.4812	2.8873	4.2526	4.7507
Cheapest Insertion + 2-opt	9.0931	5.2145	8.9455	10.6561	11.5563
Farthest Insertion + 2-opt	4.9245	1.5299	4.1329	6.1041	7.9312
RouteRepair-Constructive	16.0760	12.2957	14.5876	17.3112	20.1095
RouteRepair + 2-opt	2.8805	1.1689	2.3272	3.3359	4.6899

Interpretation. The pure generated constructor (16.0760%) is not competitive with farthest insertion (6.0049%), so constructive quality alone does not explain the final result. After identical 2-opt postprocessing, RouteRepair + 2-opt (2.8805%) is best at every scale, indicating that the learned rule and the deterministic postprocessor are complementary.

Table 7: Complete 30-instance TSPLIB transfer pools.

Task	Method	Gap (%)	Runtime (s)
GLS	RouteRepair-GLS	1.8742	18.7226
GLS	Standard GLS	2.0492	12.0822
GLS	Low-penalty GLS	2.0505	12.2042
GLS	NN + 2-opt	4.5336	1.9325
Constructive + 2-opt	RouteRepair + 2-opt	3.9244	2.6464
Constructive + 2-opt	NN + 2-opt	4.3187	2.1779
Constructive + 2-opt	Farthest Insertion + 2-opt	7.0208	1.6881

Interpretation. On the complete 30-instance TSPLIB pool, RouteRepair-GLS (1.8742%) improves on standard GLS (2.0492%) and low-penalty GLS (2.0505%). The constructive family transfers less well: RouteRepair + 2-opt (3.9244%) is close to NN + 2-opt (4.3187%), reflecting the different problem regimes covered by the pool.

Interpretation. RouteRepair's prior improves the equal-scale TSP20–TSP200 gap from 5.341% (ReEvo) to 5.145% and the weighted gap from 6.829% to 6.688%, while remaining slightly worse than ReEvo on TSP20, TSP200, and the TSP500 transfer scale. This mixed pattern is retained as measured rather than smoothed over.

Interpretation. RouteRepair attains the lowest held-out cost (26,723.6), the smallest gap to OR-Tools (17.523%), and the best CVRPLIB gap (23.001%). The EoH-adapted rule (31,042.2) is the weakest comparator, and the ReEvo-adapted guide (27,286.6) is the closest one.

Interpretation. RouteRepair reduces mean cost at every scale, from 1.88% (CVRP100) to 5.74% (CVRP200), and retains a 3.59% gain on the CVRP1000 transfer scale. Vehicle counts and utilization are unchanged by design, so the gain arises from edge ordering within the fixed feasible construction.

Table 8: TSP-ACO fixed-prior comparison (gap, %).

Method	TSP20	TSP50	TSP100	TSP200	20–200	TSP500	Weighted
RouteRepair	0.354	1.918	7.193	11.114	5.145	19.032	6.688
ReEvo	0.335	2.172	8.726	10.130	5.341	18.732	6.829
Inverse distance	0.217	5.107	13.376	21.927	10.157	37.069	13.147
Nearest rank	0.401	4.955	11.829	16.809	8.498	28.862	10.761

Table 9: Constructive CVRP under the common next-customer interface.

Method	Held-out cost	Gap to OR-Tools (%)	CVRPLIB gap (%)
Savings	27,243.6	19.744	24.835
EoH-adapted	31,042.2	35.531	34.434
ReEvo-adapted	27,286.6	20.559	23.144
Best initial LLM	28,969.4	26.034	26.845
RouteRepair	26,723.6	17.523	23.001

Interpretation. Removing expert-role separation raises the scaffold gap (0.9285%) but improves the TSP500 gap (2.1866% vs 2.3516%), a non-monotonic effect that is reported as measured. Removing memory, repair, or failure attribution each raises the gap to roughly 1.70–1.75%. Removing repair collapses LLM calls to 8, confirming that the repair loop is the dominant source of search cost.

Interpretation. Constructive CVRP repairs 253/256 instances (98.8%) with no unchanged cases and 1.2% degraded. TSP-GLS improves 158/256 (61.7%) with 7.8% degraded, and the ACO adapters retain the largest degraded fractions (12.5%), consistent with the smaller margins in those comparisons.

E Prompt and Report Structure

The diagnosis prompt explicitly separates observed evidence from inferred mechanism, names the fixed solver boundary, and prohibits code generation. The structured report then freezes the evidence, weak behavior, root-cause hypothesis, repair objective, risks, and distinct repair families. The repair prompt receives this frozen brief and is prohibited from redefining the target after seeing a candidate edit. Figure 5 shows a representative diagnosis prompt, Figure 6 shows the structured report together with the risk review and frozen brief, and Figure 7 shows a representative repair prompt.

F Additional Result Figures

Figure 8 reports constructive TSP before and after identical 2-opt postprocessing. Figure 9 and Figure 10 show TSPLIB transfer comparisons, Figure 11 and Figure 12 show the CVRP results, and Figure 13 shows the scale-wise TSP-GLS ablation.

Interpretation. Before 2-opt, the raw constructors span a wide gap range; after identical 2-opt, the gaps collapse and RouteRepair + 2-opt is best at every scale.

Interpretation. The representative instances mirror the aggregate TSPLIB comparison: RouteRepair-GLS improves on standard GLS within the same interface family, while the constructive variants transfer less well.

Interpretation. Against standard GLS, RouteRepair-GLS wins 19, ties 1, and loses 10 of the 30 TSPLIB instances; against NN + 2-opt, RouteRepair + 2-opt wins 19 and loses 11. Both distributions are retained without filtering.

Interpretation. RouteRepair attains the lowest mean cost at every held-out scale, and the EoH-adapted guide is the weakest comparator at every scale.

Table 10: CVRP-ACO comparison with matched hand-designed priors.

Scale	n	Prior	RouteRepair	Baseline	Reduction (%)
50	4	capacity-aware	10,733.5	11,018.9	2.51
100	4	capacity-aware	16,410.2	16,764.1	1.88
200	4	capacity-aware	29,970.5	31,792.3	5.74
500	4	capacity-aware	55,625.6	57,436.8	3.14
50–500	16	capacity-aware	28,184.9	29,253.0	3.32
1000	4	inverse-distance	100,039.1	103,785.8	3.59

Table 11: Overall TSP-GLS ablation results with generalization and search-cost columns. The test-gap column repeats the scaffold comparison from Table 4; TSP500 reports the same run on 64 held-out transfer instances; LLM calls and total tokens measure search cost rather than solution quality.

Ablation	Avg. gap (%)	TSP500 gap (%)	LLM calls	Total tokens
Full RouteRepair	0.7587	2.3516	176	1,432,015
w/o expert-role separation	0.9285	2.1866	128	1,122,526
w/o memory	1.7040	4.8141	176	1,420,114
w/o repair	1.7385	4.8855	8	28,630
w/o failure attribution	1.7469	4.8366	176	1,502,161

Interpretation. The overall held-out panel (96.68 vs 100.00, a 3.32% reduction) matches the aggregate CVRP-ACO comparison, and the scale-wise panel shows reductions at every scale including the CVRP1000 transfer scale.

Interpretation. The scale-wise breakdown accompanies the aggregate ablation results; separations grow on larger scales, matching the mechanism-oriented conclusion in the main text.

G Failure Modes and Interpretation Boundaries

The following boundaries are part of the intended interpretation. A verified repair is not synonymous with global dominance; it satisfies the declared recovery and collateral thresholds for one parent and one evidence-defined target. A child admitted to the evolutionary population is not automatically a verified repair. Failure cases are parent-specific rather than globally hard, so two parents may receive different diagnoses on the same dataset. Matched comparisons isolate the editable component but do not reproduce complete external systems. Finally, memory retains failed interventions because deleting them would bias later prompts toward an unrealistically positive history.

The present results also motivate additional evaluation. Repeated independent LLM campaigns would quantify search variance beyond the controlled ablation seed. Additional model families would separate framework effects from provider-specific behavior. Richer routing constraints would test whether compact static interfaces remain sufficient when time windows, stochastic demand, or dynamic dispatch state become central. These are future extensions rather than claims supported by the current experiments.

H Anonymous Code Package

The supplementary code archive exposes exactly the five adapters evaluated in the paper. It includes the fixed routing backbones, shared adapter-evolution engine, prompt templates, campaign configuration, and smoke tests. It excludes credentials, cached API responses, historical outputs, manuscripts, author metadata, unrelated problem families, and later parameter-training or neural-checkpoint experiments. Formal datasets and external benchmark executables are not redistributed; the README states how locally generated smoke data and external references are connected to the evaluation code.

Table 12: Instance-level failure-repair outcomes (percentage columns).

Adapter	Improved/tested	Improved	Unchanged	Degraded
TSP-GLS	158/256	61.7	30.5	7.8
Constructive CVRP	253/256	98.8	0.0	1.2
TSP-ACO	13/16	81.2	6.2	12.5
CVRP-ACO	14/16	87.5	0.0	12.5

Prompt for Failure Diagnosis Expert (TSP-GLS Example)

[Shared expert instruction]
Diagnose why the current parent performs poorly on its selected failure cases while identifying the behavior that should be preserved on its strength cases. Treat the root cause as a testable program-level hypothesis. Do not generate Python code.

[Problem-specific adapter: TSP-GLS]
Role: diagnose an edge-scoring function used by a fixed GLS solver.
Objective: minimize GLS-improved tour length and reference gap.
Case features: N, tour length, reference length, gap, pairwise-distance statistics, nearest-neighbor statistics, bounding-box aspect ratio, and Standard GLS comparison.

[Runtime-populated evidence]
Parent candidate: [PARENT_NAME]
Parent function: [PARENT_CODE]
Aggregate evaluation summary: [EVALUATION_SUMMARY]
Failure cases: [FAILURE_CASES] Strength cases: [STRENGTH_CASES]
Current population ranking: [ROUND_RANKING]
Baseline comparison: [BASELINE_COMPARISON]

[Shared diagnosis requirements]
Return six sections: Failure Evidence, Weak Behavior, Root Cause, Repair Objective, Targeted Repair Directions, and Risk Control. Then append exactly one valid JSON block.

[Required structured diagnosis report]

```
{
  "failure_evidence": "...",
  "weak_behavior": "...",
  "root_cause": "...",
  "repair_objective": "...",
  "repair_risk": "...",
  "repair_families": [
    {"name": "...", "category": "...",
     "target_failure_pattern": "...",
     "code_change_focus": "...", "why_distinct": "..."}
  ]
}
```

The report must contain exactly five distinct repair families: targeted patch, hybrid recombination, structural rewrite, counterfactual test, and stagnation-escape mutation.

Figure 5: Representative TSP-GLS Failure Diagnosis Expert prompt.

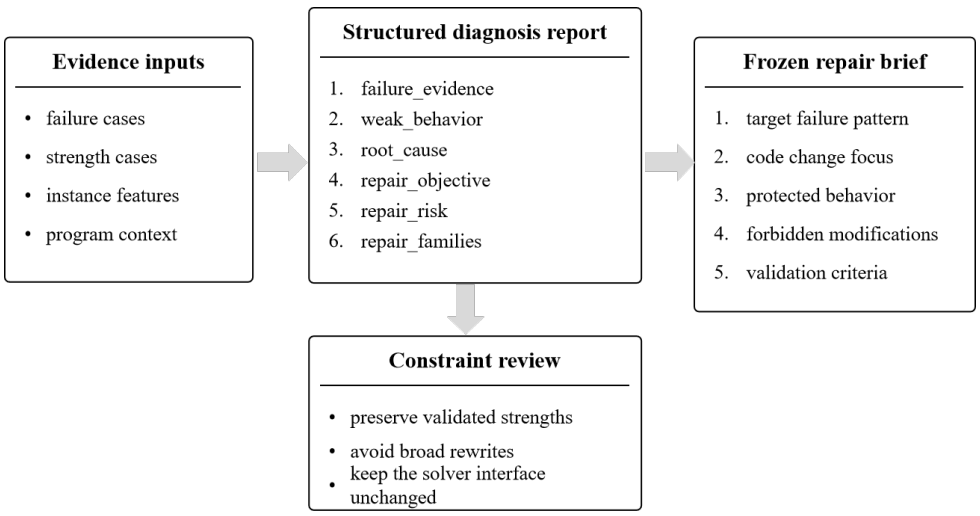


Figure 6: Structured diagnosis report, risk review, and frozen repair brief.

Prompt for Targeted Repair Expert (TSP-GLS Example)

[Shared expert instruction]
Repair one parent heuristic using the frozen structured diagnosis and reflection/risk review. Preserve useful parent behavior, follow the selected repair family and assigned repair mode, and return one executable function. Repair success is determined only by external evaluation.

[Problem-specific adapter: TSP-GLS]
Generated function: score_edge_for_penalty(city_a, city_b, path, distance_matrix, penalties, coords, state).
Return contract: one finite float; a larger value means higher GLS penalty priority.
Allowed signals: distance, penalty, coordinates, local path position, and scalar search state.
Forbidden operations: tour construction, 2-opt, full edge scan, input mutation, randomness, imports, or persistent state.

[Runtime-populated repair context]
Primary repair goal: [PRIMARY_GOAL]
Repair parent and selection reason: [REPAIR_PARENT]
Parent code: [PARENT_CODE] Evaluation summary: [EVALUATION_SUMMARY]
Failure cases: [FAILURE_CASES] Strength cases: [STRENGTH_CASES]
Elite context: [ELITE_CONTEXT]
Structured diagnosis: [DIAGNOSIS_JSON]
Reflection/risk review: [REFLECTION_RESULT]
Selected repair family: [SELECTED_REPAIR_FAMILY]
Assigned repair mode: [REPAIR_MODE]
Stagnation summary: [STAGNATION_SUMMARY]
Recent memory: [RECENT_MEMORY]

[Shared repair requirements]
The selected repair family is the primary modification direction. Non-patch modes must alter behavior beyond coefficient tuning. Elite mechanisms may inspire the repair but must not be copied verbatim. Use reflection to protect strength cases and memory only as evidence.

[Required output]
A brief statement of the intended modification followed by exactly one fenced Python code block containing one repaired function with the original signature and no imports.

The same repair template is used for all tasks; only the purple adapter block and orange runtime evidence are replaced.

Figure 7: Representative Targeted Repair Expert prompt.

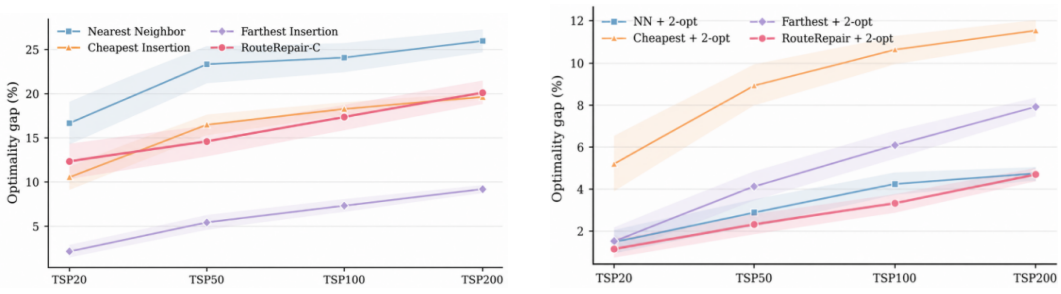


Figure 8: Constructive TSP before (left) and after (right) identical 2-opt postprocessing.

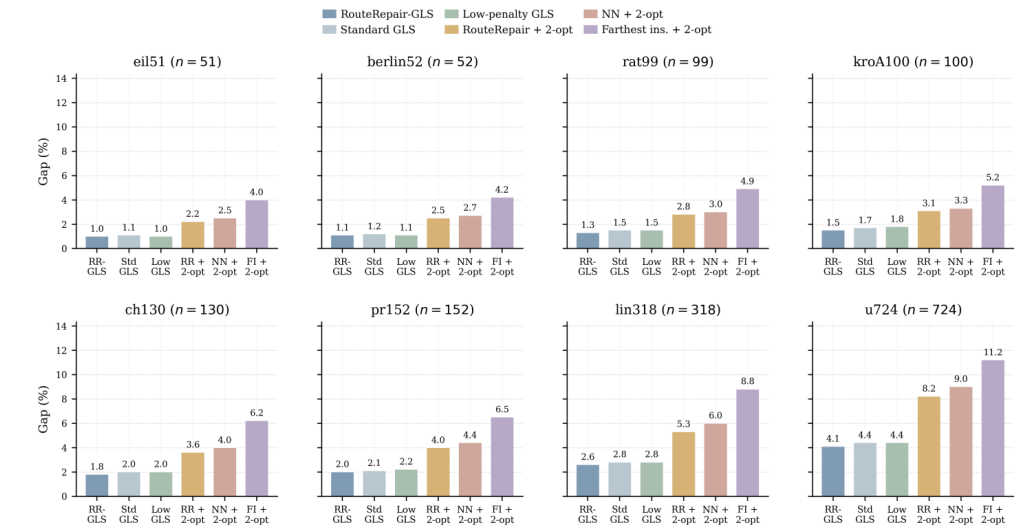


Figure 9: Representative TSPLIB transfer comparisons. Aggregate claims use the complete 30-instance pools.

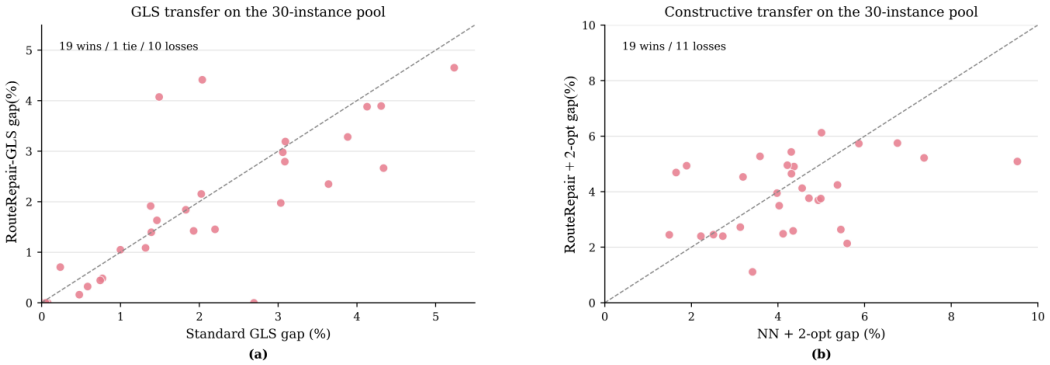


Figure 10: Paired instance-level TSPLIB transfer comparison.

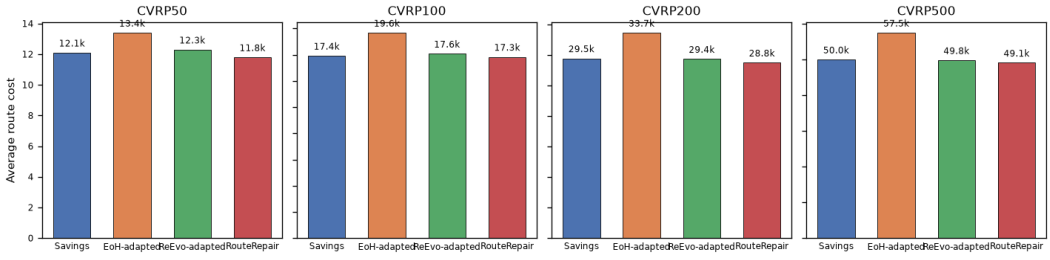
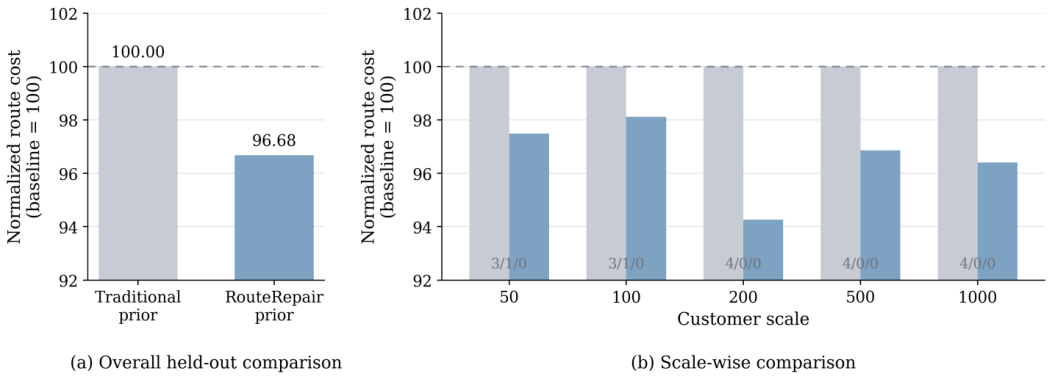


Figure 11: Scale-wise constructive CVRP results under the common next-customer interface.



(a) Overall held-out comparison (b) Scale-wise comparison

Figure 12: CVRP-ACO held-out and CVRP1000 transfer comparisons with matched priors.

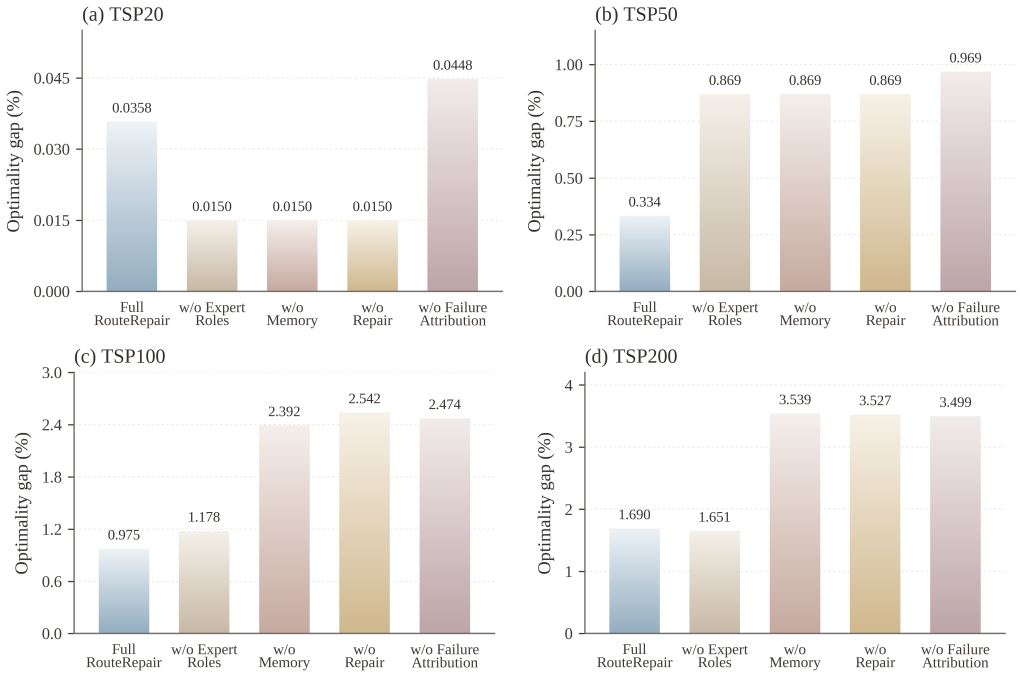


Figure 13: Scale-wise TSP-GLS ablation results.