GEPA-TSP: Specializing Lin–Kernighan Heuristics to Target Instance Distributions

Anonymous Authors[1]

Abstract

We study whether a lightweight, distribution-aware specialization loop can improve Concorde's Lin–Kernighan (LK) heuristic on specific TSP workloads. Using GEPA, an LLM-guided program search, we inject candidate LK blocks into a sandboxed Concorde build and benchmark them against held-out splits. On a non-Euclidean Seattle travel-time distribution (400 nodes), GEPA discovers a buffering/flush policy that reduces average wall time by ˜4% versus the baseline LK, while maintaining zero failures/timeouts. On two Euclidean benchmarks (uniform and clustered 400-node instances), the same candidate regresses by ˜2–3%, highlighting that gains are distribution-specific and that the tuned baseline remains strong on its native domain. We release code, datasets, and all candidate artifacts to support reproducibility and future per-distribution tuning.

1. Introduction

LLM-guided program search has emerged as a practical tool for adapting classical solvers to particular workloads. We focus on Concorde's Lin–Kernighan (LK) heuristic and ask: can we specialize LK to a target TSP distribution (e.g., Seattle travel-time vs. Euclidean) without hand-engineering? We pair GEPA's reflective mutation loop with a sandboxed Concorde pipeline that rebuilds and benchmarks candidate LK blocks on controlled splits.

Our contributions: (i) a reproducible sandbox for LK candidate injection with per-run binary hashes, CPU pinning, and artifact logging; (ii) curated splits spanning non-Euclidean (Seattle travel-time) and Euclidean (uniform, clustered) regimes; (iii) empirical evidence that specialization is distribution-dependent—GEPA finds a modest speedup (˜4%) on Seattle but regresses on Euclidean sets where the baseline is already tuned; (iv) release of code, data, and all candidate blocks to enable downstream per-distribution tuning.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2. Related Work

- **Classical LK and Concorde.** Foundational heuristics date to Lin–Kernighan's effective local search for TSP (Lin & Kernighan, 1973); Concorde's implementation and engineering remain the reference standard (Applegate et al., 2006).
- **Learning to optimize solvers.** A growing line of work learns heuristics or policies for combinatorial optimization; our setting follows the same spirit but targets distribution-specific LK tweaks.
- **LLM-guided code evolution.** ReEvo frames LLMs as reflective hyper-heuristics that iteratively refine algorithms (Ye et al., 2024); our GEPA loop similarly mutates and tests LK code but with a sandboxed, deterministic TSP pipeline.

3. Method: GEPA for Lin–Kernighan

- Sandbox: copy Concorde, inject LK block between sentinel markers, rebuild in isolation, and run scripted evals on a chosen split.
- Metric: negative average wall time (primary); we log BB nodes, timeouts, and failures; runs are cached with binary SHA256 and CPU affinity for reproducibility.
- Prompts: student emits a replacement LK block; reflector proposes edits; optional overrides steer toward buffering/flush policies.
- Safety: ANSI C89, no globals or I/O, bounded buffers; dedup guard to avoid re-evaluating identical blocks.
- Workflow: pick a target split, run GEPA for a small budget (20 steps), archive all artifacts (code, logs, metrics).

4. Benchmarks and Data

- Non-Euclidean: structured_seattle_time (400 nodes) from OSM travel-time shortest paths (val/test splits of 20/50 instances).
- Euclidean: uniform_val/test (400 nodes) and clustered_val/test (400 nodes); metadata and seeds released.
- Other splits (toy20/200, tsplib_random) maintained for smoke/regression; not central to main findings.

5. Experimental Setup

- Models: student gpt-5-nano, reflector gpt-5-mini; reflection batch 2–3; 20 metric calls.
- Evaluation: per-instance repeats (3–5 on val), CPU affinity when available, timeouts off for reported runs; artifacts under `runs/`.
- Baseline: Concorde default LK rebuilt in the same sandbox; baseline repeats higher (5) for a stable reference.
- Variance: report per-instance averages; note that nontrivial gains require reproducible settings (affinity, binary hash).

6. Results: TSP Adaptation

- **Uniform (Euclidean, test 50 inst.):** baseline runtime 4.223 s / BB 3.84 vs GEPA best 4.521 s / BB 3.96 (+7.1% runtime, +3.1% BB). Plots: `out/gepa_uniform_n400_mean_std.png`; summaries: `runs/eval/eval/20251202T224329Z_uniform_test_baseline`, `runs/eval/eval/20251202T225646Z_uniform_test_gepa_iter40`.
- **Clustered (Euclidean, test 50 inst.):** baseline 4.405 s / BB 4.6 vs GEPA 4.544 s / BB 5.2 (+3.1% runtime, +13.0% BB). Plots: `out/gepa_clustered_20251129T211605Z.png`; summaries: `runs/eval/eval/20251202T230700Z_clustered_test_baseline`, `runs/eval/eval/20251202T231054Z_clustered_test_gepa_iter30`.
- **Seattle (travel-time, test 50 inst.):** baseline 3.958 s / BB 3.68 vs GEPA 3.768 s / BB 3.55 (−4.8% runtime, −3.5% BB); steady improvement in the rollout (see `out/gepa_structured_seattle_time_n400_time_smoothed_final.png`). Summaries: `runs/eval/eval/20251202T231521Z_seattle_time_test_baseline` (latest `runs/eval/eval/20251202T233939Z_seattle_time_test_baseline_latest`) and `runs/eval/eval/20251202T231850Z_seattle_time_test_gepa_iter31`.
- Overall: GEPA slows Euclidean cases relative to the tuned baseline, but yields a modest Seattle speedup while slightly reducing BB nodes, underscoring distribution-specific effects.

7. Discussion

- GEPA excels when the baseline is not already tuned: non-Euclidean Seattle benefits; tuned Euclidean baselines do not.
- The learned tweak targets buffer/flush overhead; it may hurt when coherent batches are valuable (Euclidean).
- Reproducibility is essential: affinity, binary hashes, and artifact logging prevent confounding from build drift.
- Deployment: treat GEPA as per-distribution autotuning—run briefly on your workload, adopt the candidate if it beats your baseline.

8. Conclusion

- GEPA can specialize LK for specific TSP distributions, yielding modest gains on non-Euclidean travel-time data while leaving tuned Euclidean baselines unchanged or slightly worse.
- Future work: multi-objective rewards, better diversity in proposals, and extensions beyond TSP.
- Release: code, data, and all candidate artifacts for reproducibility.

References

Applegate, D. L., Bixby, R. E., Chvátal, V., and Cook, W. J. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

Lin, S. and Kernighan, B. W. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973. doi: 10.1287/opre.21.2.498.

Ye, H., Wang, J., Cao, Z., Berto, F., Hua, C., Kim, H., Park, J., and Song, G. Reevo: Large language models as hyper-heuristics with reflective evolution. *arXiv preprint arXiv:2402.01145*, feb 2024. URL https://arxiv.org/abs/2402.01145. NeurIPS 2024.