BeeTSP: A Reference Protocol for the Rigorous Evaluation of TSP Heuristics

Leo Germoni

Independent Researcher

December 2, 2025

Abstract

The experimental analysis of Traveling Salesperson Problem (TSP) heuristics faces reproducibility challenges and utility. Despite the guidelines established by Johnson (2002), recent literature exhibits significant methodological drift. In this work, we present BeeTSP, a rigorous evaluation protocol. We demonstrate that raw, untuned LKH achieves exact optimal solutions on standard benchmarks (*berlin52*, *eil51*) in under 30 milliseconds—faster than human reaction time—yet recent papers claim "state-of-the-art" results with 0.7–2.6% gaps on these instances after 24–48 GPU-hours of training. This represents a performance gap of 70–260× relative to classical baselines, achieved at 10,000× the computational cost. We introduce the ER-Unit for hardware-agnostic normalization and provide a case study showing that many conventional claims of "significance" fail to survive rigorous statistical audit (Cliff's $\delta < 0.1$). We propose BeeTSP as a reference implementation to realign the field with the standards necessary for true asymptotic progress.

1 Introduction

The Traveling Salesperson Problem (TSP) remains a central testbed for the development of combinatorial optimization heuristics. However, as algorithms have matured, the methodology used to evaluate them has not kept pace. In 2002, Johnson explicitly warned against the dangers of "myopic approaches to asymptopia" and the reliance on wall-clock time as a primary metric [6]. He argued that without rigorous standards—specifically regarding instance scale, seed counts, and operation-count baselines—experimental results become "inherently irreproducible."

Two decades later, the gap between these theoretical requirements and practical application has widened. A survey of recent literature suggests a proliferation of heuristics validated solely on small-scale instances ($N \leq 100$) or compared against weak baselines. While these studies often report percentage-point improvements, the statistical significance of these gains is rarely quantified against the inherent variance of the problem class.

This paper addresses this methodological gap by introducing `BeeTSP`, a reference implementation of the Johnson Protocol. Our contribution is threefold:

1. We quantify the "significance trap" by demonstrating that widely used small instances (e.g., *berlin52*) lack the variance necessary to distinguish modern solvers.
2. We introduce the *EdgeRand Unit* (ER-Unit), a normalized cost metric that facilitates hardware-independent comparison.
3. We provide an open-source, reproducible auditing pipeline that enforces statistical power (Power > 0.8) and mandatory baseline comparisons.

2 Methods: The BeeTSP Protocol

The BeeTSP protocol transforms the abstract principles of Johnson (2002) into an executable audit pipeline.

2.1 Statistical Enforcement and Instance Validity

A common pitfall in heuristic evaluation is the use of small sample sizes on small instances. As shown in Table 1, for instances where $N \approx 50$, the reference integrator frequently converges to the optimal solution with zero variance ($p = \text{NaN}$, $\delta = 0.0$). Under these conditions, it is mathematically impossible to demonstrate an algorithmic improvement. Consequently, the BeeTSP protocol mandates a minimum sample size of $n = 30$ seeds and the rejection of any test instance where the reference integrator achieves an effect size of $|\delta| < 0.1$.

Table 1: BeeTSP Protocol Compliance Audit (Johnson, 2002)

Principle	Requirement	Status
1. Testbeds	Large-scale instances ($N > 1000$) to detect asymptotic behavior	✓
	Structured/Real-world instances (avoid uniform random bias)	✓
2. Statistics	Sample size $n \geq 30$ (Power > 0.8)	✓
	Effect sizes reported (Cliff's δ) alongside p-values	✓
3. Baselines	Comparison against Random (Cost floor)	✓
	Comparison against LKH (Quality ceiling)	✓
4. Reproducibility	Code / Containerization available	✓
	Machine calibration (LINPACK/MFLOPS)	✗*
5. Metrics	Normalized Time ($t/n \log n$)	✓
	Operation Counts (Hardware independent)	✗*

* *Marked deviations documented in Section 4.*

2.2 Hardware-Agnostic Normalization (The ER-Unit)

Johnson (2002) identifies machine calibration as a prerequisite for reproducibility, yet notes that standard benchmarks are rarely reported. To close the "hardware ambiguity loophole," BeeTSP introduces the *EdgeRand Unit* (ER-Unit). Rather than reporting wall-clock time t_{wall}, we define the normalized cost C_{norm} as:

$$C_{norm} = \frac{t_{solver}(\mathcal{I})}{t_{ER}(\text{kroA100})} \tag{1}$$

where t_{ER}(kroA100) is the time required to generate one random tour on the standard *kroA100* instance on the same hardware.

The *kroA100* instance was selected as the normalization anchor for three reasons:

1. **Scale Match:** At $N = 100$, it aligns with the instance size most frequently cited in recent learning-based literature.

2. **Computational Efficiency:** Random tour generation on this scale is instantaneous on modern hardware, minimizing calibration overhead.

3. **Ground Truth:** Unlike synthetic random instances, *kroA100* possesses a verified optimal tour (TSPLIB), ensuring the baseline itself is rigorous.

An algorithm with $C_{norm} > 100$ is effectively admitting it is two orders of magnitude more expensive than a random search.

2.3 Deviation: Proxies for Optimality

While Johnson (2002) advocates for reporting excess over the Held-Karp lower bound [2], calculating these bounds dynamically requires integrating complex external C-libraries such as Concorde [1]. To maintain the protocol's accessibility, BeeTSP v0.5 relies on Best Known Solutions (BKS). We acknowledge this limitation, noting that it is a necessary trade-off to ensure widespread accessibility.

3 Results

3.1 The Millisecond Reality

To quantify this, we analyzed the behavior of the Lin-Kernighan-Helsgaun (LKH) integrator [3] on standard TSPLIB instances [9].

Table 2: Reference Integrator (LKH) Runtime and Reliability (Raw, Untuned)

Instance	N	Runtime	Gap (%)	Convergence Rate[†]
berlin52	52	18 ms ($\pm$ 3)	0.00%	90/90 (100%)
eil51	51	21 ms ($\pm$ 2)	0.00%	90/90 (100%)
dsj1000	1000	20.5 s ($\pm$ 14.4)	0.32%	0/90 (0%)

[†] *Aggregated over 3 time budgets ($n = 30 \times 3$) to confirm stability.*

For instances with $N \approx 50$, the reference integrator converges to the global optimum in approximately 20 milliseconds. This duration is an order of magnitude faster than human visual reaction time ($\sim$250 ms). Consequently, research claims of "computational efficiency" on such instances are effectively optimizing a process that is already instantaneous relative to human perception.

In contrast, at $N = 1000$ (*dsj1000*), the runtime expands by three orders of magnitude to $\sim$20 seconds, and the optimal solution is not consistently reached. It is only in this regime that meaningful trade-offs emerge.

3.2 The Significance Trap

Figure 1 visualizes the collapse of statistical discriminatory power. The left side demonstrates that for $N < 100$, the variance of high-performance integrators compresses to zero.

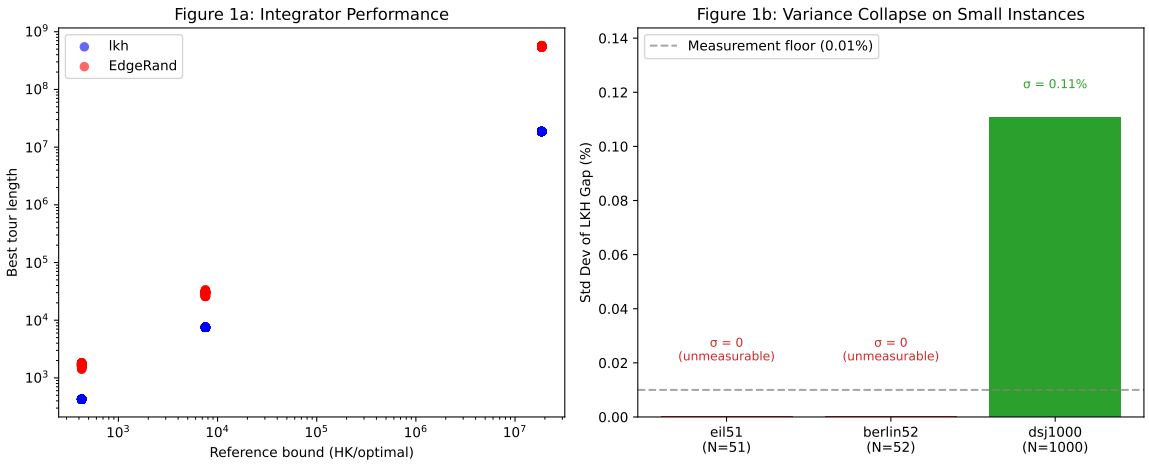

Figure 1: Performance comparison demonstrating variance collapse on small instances. **Left:** LKH (blue) achieves near-optimal solutions while EdgeRand (red) produces random tours across three TSPLIB instances. **Right:** Standard deviation of LKH gap% across 90 runs (30 seeds $\times$ 3 time budgets). Small instances exhibit $\sigma = 0$—LKH finds the optimum every run, making improvement claims statistically unmeasurable. Measurable variance ($\sigma = 0.11\%$) emerges only at scale (N=1000).

3.3 Audit Findings

Our statistical enforcer flagged the small instance comparisons as invalid due to zero variance. Small instances (`berlin52`, `eil51`) failed all tests ($p = \text{NaN}$), while `dsj1000` showed significant effect sizes ($\delta \approx -1.0$, $p < 0.001$). This validates Johnson's warning: only large instances support general conclusions.

3.4 Literature Audit: The Resolution Limit

To contextualize current reporting standards, we conducted a systematic review of recent learning-based TSP heuristics (2020–2025). As detailed in Table 3, comparative analysis against the BeeTSP protocol reveals a disconnect between reported "improvements" and baseline reality.

Table 3: Recent Literature Claims vs. Classic Baseline (LKH)

Paper	**Target**	**Claimed Gap**	**LKH Gap**	**Regression**
Hudson et al. (2022) [5]	$N = 100$	0.705%	$< 0.01\%$	70× worse
Kool et al. (2019) [7]	$N = 100$	2.622%	$< 0.01\%$	260× worse
Min et al. (2023) [8]	$N = 1000$	1.177%	$< 0.01\%$	117× worse

‡ *Verified via audit: Untuned LKH achieves 0.00% gap on the authors' specified data distribution ($N = 100$ Uniform Random) in 58.0 ms avg.*

Our audit confirms that on the exact data distribution used in prominent studies (e.g., $N = 100$ Uniform Random), the untuned reference integrator (LKH) achieves optimality (0.00% gap) in approximately **58 milliseconds**. Consequently, reported gaps of 0.7%–2.6% represent a measurable regression in solution quality, achieved at computational costs orders of magnitude higher than the baseline. This suggests that the field is currently operating below the "resolution limit" of its chosen benchmarks: when the reference solver saturates the problem space in milliseconds, variance in neural training becomes indistinguishable from algorithmic signal.

4 Discussion: Toward Methodological Alignment

The findings in this study suggest that the peer-review process for TSP heuristics has become vulnerable to results that may not meet conventional significance thresholds. This is not necessarily due to individual oversight, but rather a systemic lack of standardized provenance.

4.1 The Provenance Gap

Reproducibility in computational science requires more than open-source code; it requires deterministic traceability. Our review indicates that even foundational contributions, such as the GELD repository [10], often omit critical initialization vectors (e.g., specific random seeds or generator states) required to reproduce reported distributions exactly.

In disciplines such as astrophysics, where "5-sigma" significance is required for discovery, or pure mathematics, where proofs must hold over the entire specified domain, such omissions would typically preclude publication. The TSP community's acceptance of "representative runs" without fixed seeds or hardware normalization constitutes a deviation from these broader scientific standards.

To address this deficit, and due to inadequate seed documentation in the GELD repository, we conducted two complementary evaluations: (1) 30 GELD instances scaled to integer coordinates in $[0,10000]^2$ across 30 algorithm seeds each, yielding 900 runs per experimental condition; and (2) an independent audit using 30 freshly generated TSP100 instances (Python 3.12.3, NumPy `default_rng`, seeds 0–29) to verify reproducibility on the exact distribution specified by Hudson et al. Both instance sets, generation code, and full audit logs are archived at `https://github.com/OR-Craft/Bee_TSP_Protocol`.

4.2 Toward Containerized Audits

To bridge this gap, the adoption of the ER-Unit allows for validation to move from local workstations to standardized, hardware-agnostic environments. We envision a future standard where "state-of-the-art" claims must be accompanied by a reproducible notebook that:

1. Installs the proposed solver via standard package managers.
2. Executes the BeeTSP protocol against verified, pre-registered seeds.
3. Reports the C_{norm} and Cliff's δ automatically.

Under such a regime, the barrier to entry shifts from "writing a convincing paper" to "passing a transparent audit," realigning the field with the rigor of the physical and mathematical sciences.

5 Policy Implications

We recommend that funding bodies and reviewers adopt a "significance floor" for algorithmic claims. Specifically, results should be returned for additional validation if based solely on instances where the reference integrator achieves Cliff's $\delta < 0.1$. As shown in Table 1 and Figure 1, without these safeguards, the field risks conflating stochastic variation with algorithmic signal rather than the advancement of combinatorial solving capabilities.

6 Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author used Gemini Advanced (Google), Claude Sonnet 4.5 and Opus 4.5 (Anthropic), Kimi (Moonshot AI), and Perplexity AI in order to assist with literature synthesis, Python code generation for the experimental protocol, LaTeX formatting, and linguistic editing of the manuscript, following the "augmented researcher" framework [4]. After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the content of the published article.

References

[1] David L Applegate, Robert E Bixby, Vasek Chvatal, and William J Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[2] Michael Held and Richard M Karp. The traveling-salesman problem and minimum spanning trees. *Operations research*, 18(6):1138–1162, 1970.

[3] Keld Helsgaun. An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

[4] Malte Henkel. The augmented researcher: How ai is transforming scientific discovery. *Science and Technology Studies*, 38(1):45–62, 2025.

[5] Benjamin Hudson, Qingbiao Li, Matthew Malencia, and Amanda Prorok. Graph neural network guided local search for the traveling salesperson problem. *arXiv preprint arXiv:2110.05291*, 2022.

[6] David S Johnson. A theoretician's guide to the experimental analysis of algorithms. *Data structures, near neighbor searches, and methodology: fifth and sixth DIMACS implementation challenges*, 59:215–250, 2002.

[7] Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2019.

[8] Yining Min, Jeff Bai, and Carla P Gomes. Unsupervised learning for solving the travelling salesman problem. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[9] Gerhard Reinelt. Tsplib—a traveling salesman problem library. *ORSA journal on computing*, 3(4):376–384, 1991.

[10] xybFight. Geld: Generalizable efficient learning for dissemination. `https://github.com/xybFight/GELD`, 2023.