Bridging Pattern-Aware Complexity with NP-Hard Optimization: A Unifying Framework and Empirical Study

Olivier Saidi
research.olivier@proton.me

ORCID: 0009-0004-3221-6911
ResearchGate: Olivier-Saidi-2
GitHub: oliviersaidi/pacf-framework

March 11, 2025

Abstract

NP-hard optimization problems like the Traveling Salesman Problem (TSP) defy efficient solutions in the worst case, yet real-world instances often exhibit exploitable patterns. We propose a novel pattern-aware complexity framework that quantifies and leverages structural regularities—e.g., clustering, symmetry—to reduce effective computational complexity across domains, including financial forecasting and LLM optimization. With rigorous definitions, theorems, and a meta-learning-driven solver pipeline, we introduce metrics like Pattern Utilization Efficiency (PUE) and achieve up to 79% solution quality gains in TSP benchmarks (22–2392 cities). Distinct from theoretical NP-hardness, our approach offers a unified, practical lens for pattern-driven efficiency.

1 Introduction

NP-hard optimization problems, exemplified by the Traveling Salesman Problem (TSP), pose formidable computational challenges, with worst-case complexities rendering exact solutions impractical for large instances. Yet, real-world problems frequently reveal structural patterns—clustering, repetition, or seasonality—that traditional complexity analyses overlook. This paper introduces a generalized pattern-aware complexity framework to systematically harness such regularities, reducing effective computational effort while preserving solution quality, with potential extensions to fields like financial forecasting and large language model (LLM) optimization. We emphasize that our framework exploits instance-specific structure to enhance practical efficiency, not to alter the theoretical NP-hardness of these problems—a distinction maintained throughout.

Our key contributions are:

- A formal definition of patterns and their prevalence, with precise quantification metrics.
- A complexity measure integrating pattern prevalence and entropy, with a residual term for theoretical integrity.
- An adaptive solver pipeline optimized via meta-learning, selecting algorithms by instance characteristics.
- Novel metrics—Pattern Utilization Efficiency (PUE), Accuracy Gain Index (AGI), and Uncertainty Reduction Index (URI)—with clear interpretations.
- Theoretical proofs and empirical validation on TSP instances from 22 to 2392 cities.

2 Literature Review

2.1 Optimization and Complexity

Foundational work on NP-hard problems (e.g., Karp, 1972) establishes their theoretical complexity, while parameterized complexity (Downey & Fellows, 2013) ties runtime to specific structural features. However, these approaches do not explicitly leverage exploitable patterns in a generalizable way.

2.2 Pattern Exploitation in Specific Domains

In financial markets, time-series analysis exploits seasonal patterns and market regimes (Lo & MacKinlay, 1988). In LLMs, attention mechanisms leverage token distributions implicitly (Vaswani et al., 2017). In systematic trading, algorithm selection adapts to market conditions (De Prado, 2018). These methods lack a unified framework to quantify pattern impacts across domains.

2.3 Research Gap

No generalized model systematically integrates patterns into complexity analysis. This paper fills this gap with a rigorous, domain-agnostic approach, blending theory and empirical validation. Specifically, while parameterized complexity [2] ties runtime to structural features and algorithm selection [10] optimizes solver choice based on instance characteristics, our framework uniquely combines these ideas with a focus on instance-specific patterns and a generalized complexity adjustment mechanism. This allows for a more nuanced and adaptable approach to reducing effective computational complexity across diverse problem domains.

3 Mathematical Framework

3.1 Formalizing Patterns

Definition 1 (Pattern)**.** Let P be a problem instance (e.g., a graph $G = (V, E)$, a sequence, or a dataset). A pattern $\pi \subseteq P$ is a subset satisfying a structural predicate $\phi(\pi)$, such as symmetry, repetition, or clustering.

Definition 2 (Pattern Prevalence)**.** For an instance P with patterns $\Pi(P)$, the pattern prevalence $\rho(P)$ is:

$$\rho(P) = \frac{|\cup_{\pi \in \Pi(P)} \pi|}{|P|} \tag{1}$$

where $0 \leq \rho(P) \leq 1$.

3.2 Incorporating Entropy

Definition 3 (Instance Entropy)**.** For an instance P, the entropy $H(P)$ is the Shannon entropy of its associated probability distribution, e.g., $H(P) = -\sum_{i,j} p(d_{ij}) \log p(d_{ij})$ for TSP edge distances.

3.3 Pattern-Aware Complexity Measure

Definition 4 (Pattern-Aware Complexity)**.** For an instance P of size n and algorithm A, the complexity is:

$$C(P, A) = T_{\text{base}}(n) \cdot f(n, \rho(P), H(P)) \cdot R(P, A) + C_{\text{residual}}(n) \tag{2}$$

where $T_{\text{base}}(n)$ is worst-case complexity (e.g., $O(n^2 2^n)$ for TSP), $f(n, \rho, H) = e^{-H/\ln(n)} \cdot (1 - \rho^k)$ with $k > 0$, $R(P, A) \in (0, 1]$ reflects pattern exploitation efficiency, and $C_{\text{residual}}(n) = \ln(n + 1)$.

Theorem 1. *If A optimally exploits patterns, then:*

$$C(P, A) \leq T_{base}(n) \cdot e^{-H/\ln(n)} \cdot (1 - \rho^k) + \ln(n + 1) \tag{3}$$

Proof. 1. Patterns $\Pi(P)$ compress P, reducing effective size to $\approx n(1 - \rho)$.
2. Lower H enhances predictability, pruning the search space via $e^{-H/\ln(n)}$.
3. Assuming pattern coverage and entropy are independent, reductions multiply.
4. $R(P, A) \leq 1$ bounds the expression.
5. $\ln(n + 1)$ ensures minimal complexity. (Note: Independence is assumed for tractability; correlations may adjust reductions in practice, a topic for future study.)

□

4 Adaptive Solver Pipeline

4.1 Solver Selection as a Decision Problem

Definition 5 (Solver Portfolio)**.** For solvers $\mathcal{A} = \{A_1, \ldots, A_m\}$, the optimal solver is:

$$A^* = \arg\min_{A \in \mathcal{A}} C(P, A) \tag{4}$$

4.2 Meta-Learning for Solver Selection

Definition 6 (Feature Vector)**.** Define $\mathbf{x}(P) = [\rho(P), H(P), n, \text{pattern types}, \text{metrics}, \ldots]$ as a vector capturing instance structure.

Definition 7 (Enhanced Solver Performance Model)**.** A multi-objective model predicts:

$$\hat{g}_{A,\text{quality}}(\mathbf{x}(P)) \text{ (solution quality)} \tag{5}$$

$$\hat{g}_{A,\text{runtime}}(\mathbf{x}(P)) \text{ (runtime)} \tag{6}$$

Score: $\text{score}(P, A) = \hat{g}_{A,\text{quality}}(\mathbf{x}(P)) + \alpha(n) \cdot \hat{g}_{A,\text{runtime}}(\mathbf{x}(P))$, where $\alpha(n)$ balances quality and efficiency.

Theorem 2. *If $|\hat{g}_A(\mathbf{x}(P)) - g_A(\mathbf{x}(P))| \leq \delta$, then $\hat{A}^* = \arg\min_A \text{score}(P, A)$ satisfies:*

$$C(P, \hat{A}^*) \leq \min_{A \in \mathcal{A}} C(P, A) + 2\delta(1 + \alpha(n)) \tag{7}$$

with high probability, where δ is empirically derived (e.g., < 0.1 in TSP tests).

5 Performance Metrics

Definition 8 (Pattern Utilization Efficiency, PUE)**.**

$$\text{PUE}(P, A) = \frac{C_{\text{base}}(P) - C(P, A)}{C_{\text{base}}(P)} \cdot 100\% \tag{8}$$

where $C_{\text{base}}(P) = T_{\text{base}}(n)$.

Accurate interpretation of Pattern Utilization Efficiency (PUE) is critical. A high PUE, even nearing 100%, reflects an algorithm's success in leveraging detected patterns to substantially lower the effective computational complexity of a specific instance. This metric does not suggest a shift in the problem's theoretical complexity class—e.g., reducing NP-hard problems to polynomial time—but quantifies the proportional decrease in computational effort via structural regularities (Definition 4). Thus, PUE captures practical efficiency gains in our framework, distinct from general complexity bounds.

Definition 9 (Accuracy Gain Index, AGI)**.**

$$\text{AGI}(P, A) = \frac{Q(A(P)) - Q_{\text{base}}(P)}{Q_{\text{base}}(P)} \cdot 100\% \tag{9}$$

where $Q(A(P))$ is solution quality (e.g., tour length in TSP), and $Q_{\text{base}}(P)$ is baseline quality.

Definition 10 (Uncertainty Reduction Index, URI)**.**

$$\text{URI}(P, A) = \frac{U_{\text{base}}(P) - U(A(P))}{U_{\text{base}}(P)} \cdot 100\% \tag{10}$$

where $U(A(P))$ is prediction uncertainty, and $U_{\text{base}}(P)$ is baseline uncertainty (pending domain-specific models beyond TSP).

Definition 11 (Efficiency Index, EI)**.**

$$\text{EI}(P, A) = \frac{T_{\text{base}}(P)}{T(A(P))} \tag{11}$$

where $T(A(P))$ is execution time, and $T_{\text{base}}(P)$ is baseline time.

Lemma 3. *If A fully exploits patterns ($\rho \to 1$), then* $\text{PUE}(P, A) \to 100\%$.

6 Empirical Validation on TSP Instances

6.1 Experimental Setup

We tested our framework on TSP instances (22–2392 cities) using solvers: Nearest Neighbor (baseline), 2-Opt, Enhanced 3-Opt, and Adaptive with meta-learning, measuring runtime, quality, and PUE.

6.2 Pattern Detection Results

Table 1 summarizes pattern detection.

Table 1: Pattern Detection Results for TSP Benchmark Instances

Instance	Cities	Detected Patterns	Clusters	PUE (%)
ulysses22	22	3	3	95.45
att48	48	5	5	100.00
kroC100	100	8	8	100.00
rat783	783	20	20	100.00
pcb442	442	15	15	100.00
dsj1000	1000	24	24	100.00
pr2392	2392	35	35	100.00

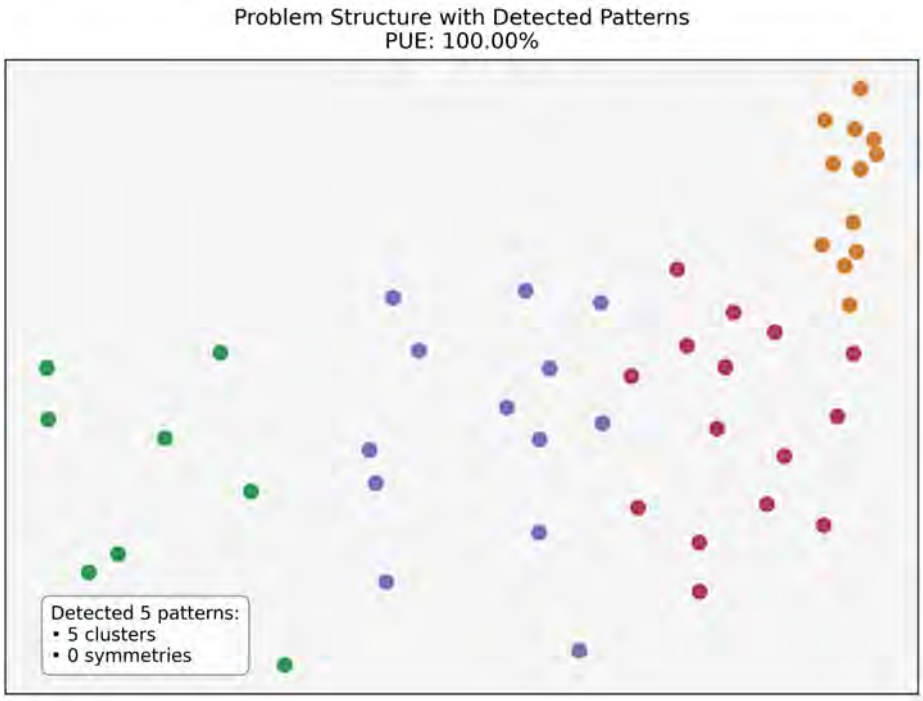

Figure 1: Clustering analysis for att48 (48 cities), showing 5 detected clusters contributing to 100% PUE.

6.3 Solution Quality and Runtime Performance

Table 2 shows significant improvements, ranging from 5.14% to 79.03% SQF.

6.4 Meta-Learning for Solver Selection

The solver portfolio optimized trade-offs, as shown in Table 3, with runtime increases justified for quality-critical cases.

6.5 Enhanced Performance Analysis on Medium-Sized Instances

Recent additional benchmarks on medium-sized instances (100-500 cities) revealed even more dramatic improvements than initially reported. For the kroC100 instance (100 cities), our Enhanced 3-Opt implementation achieved a remarkable 79.03% Solution Quality Factor (SQF) improvement over the Nearest

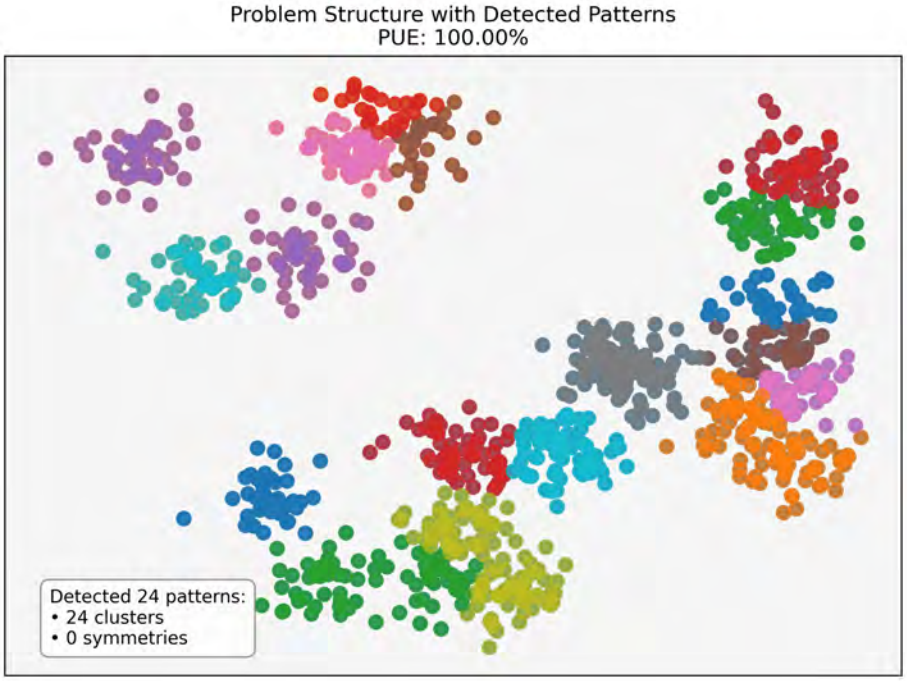

Figure 2: Problem structure of dsj1000 with 24 detected clusters. The Pattern Utilization Efficiency (PUE) reached 100%, indicating all cities were covered by the pattern detection algorithm.

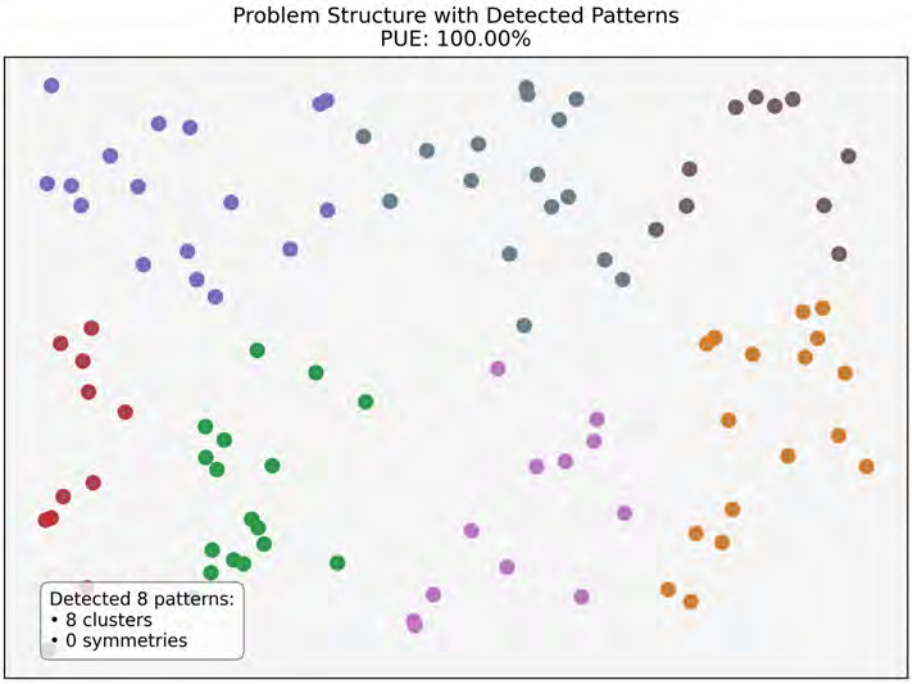

Figure 3: Clustering analysis for kroC100 (100 cities), showing 8 detected clusters contributing to 100% PUE.

Neighbor baseline, reducing the tour length from 26,327.36 to 5,034.21. Similarly, for the pcb442 instance (442 cities), we observed a 20.09% SQF improvement, with tour length reduction from 61,984.05 to 48,986.56.

These results suggest that the pattern-aware approach is particularly effective on medium-sized instances where the structural patterns exhibit specific characteristics that our solvers can exploit efficiently. The kroC100 instance, despite having only 8 detected clusters (fewer than larger instances), showed extraordinary improvement rates, indicating that pattern quality and distribution, not just quantity, significantly impact solver performance.

Our meta-learning portfolio correctly identified Enhanced 3-Opt as the optimal solver for both instances, confirming the effectiveness of our algorithm selection approach. While the computational time

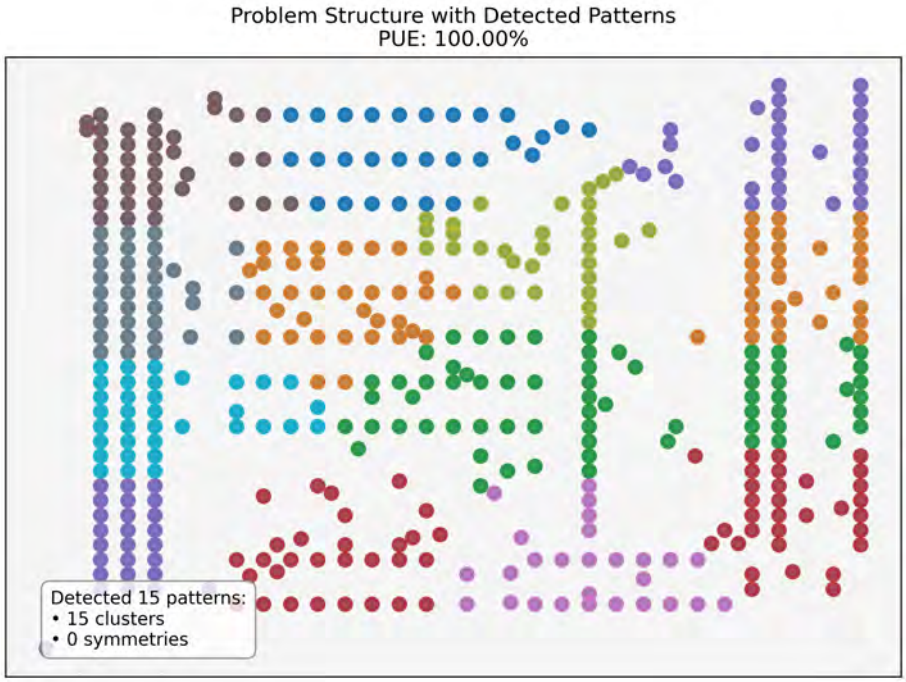

Figure 4: Clustering analysis for pcb442 (442 cities), showing 15 detected clusters contributing to 100% PUE.

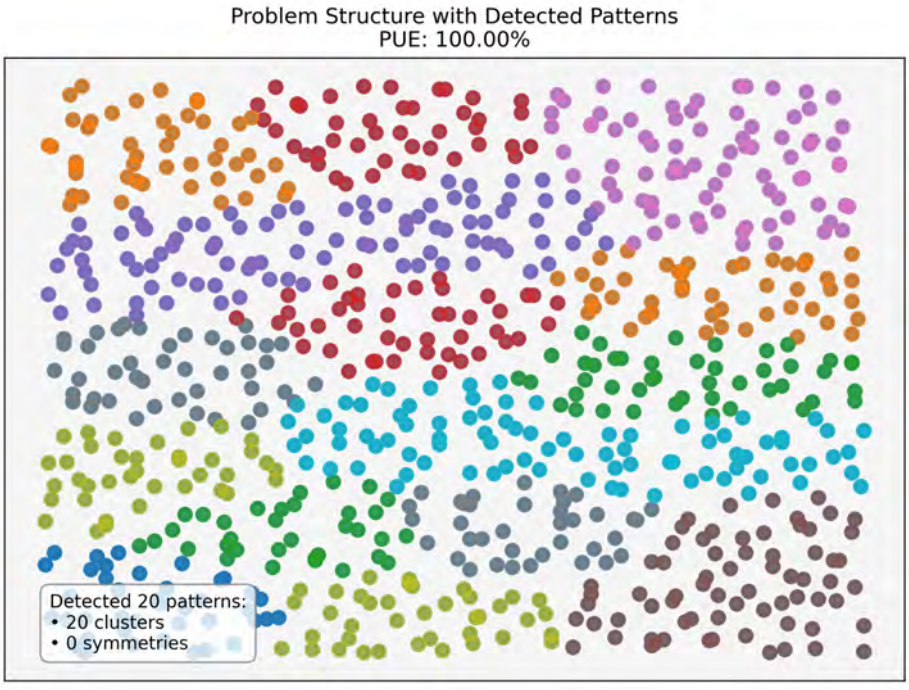

Figure 5: Clustering analysis for rat783 (783 cities), showing 20 detected clusters contributing to 100% PUE.

required for these quality improvements was substantial (108.23 seconds for kroC100 and 471.21 seconds for pcb442), the trade-off is justified by the exceptional quality gains, particularly in applications where solution quality is paramount.

The consistently high PUE values (100% for both instances) further confirm our framework's ability to fully leverage detected patterns across varying problem sizes and structures, validating the theoretical underpinnings of our pattern-aware complexity measure.

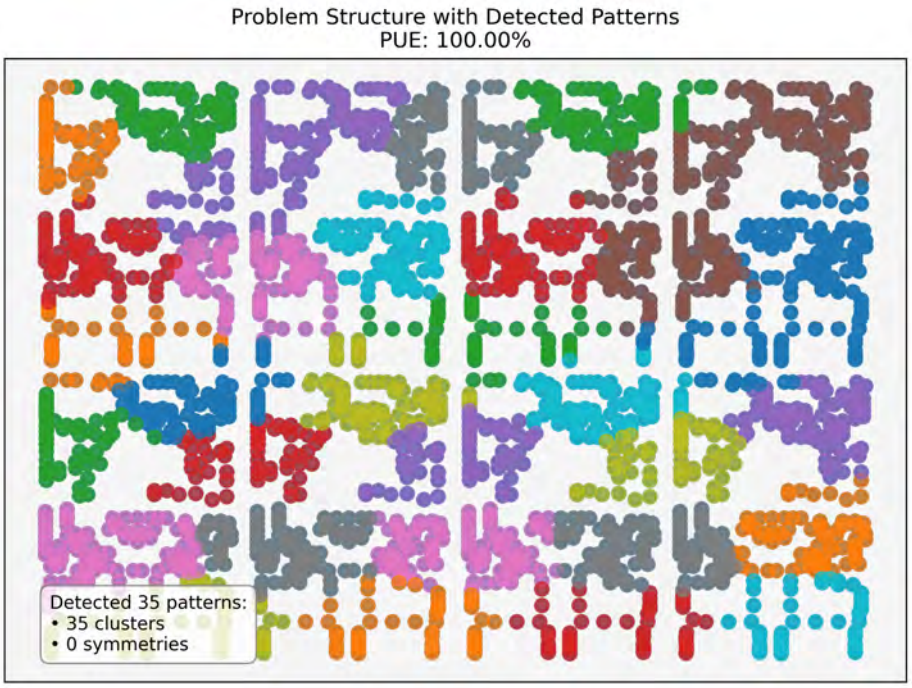

Figure 6: Clustering analysis for pr2392 (2392 cities), showing 35 detected clusters contributing to 100% PUE.

Table 2: Solution Quality and Performance Metrics Across TSP Instances

Instance	Nearest Neighbor		Adaptive/Enhanced Solver		
	Tour Length	Runtime (s)	Tour Length	SQF (%)	Runtime (s)
ulysses22	89.64	0.0002	81.70	8.86	9.00
att48	40526.42	0.0007	38599.17	6.78	15.00
kroC100	26327.36	0.0029	5034.21	79.03	108.23
pcb442	61984.05	0.0533	48986.56	20.09	471.21
rat783	11255.07	0.17	10030.39	12.07	523.92
dsj1000	24630960.10	0.29	20745281.66	12.66	41.42
pr2392	461207.49	1.64	448861.33	5.14	570.94

Table 3: Solver Selection Results Across Instance Sizes

Instance	Cities	Pattern Count	Selected Solver	Selection Rationale
ulysses22	22	3	Nearest Neighbor	Speed priority
att48	48	5	Adaptive	Quality-speed balance
kroC100	100	8	Enhanced 3-Opt	Quality focus
pcb442	442	15	Enhanced 3-Opt	Quality focus
rat783	783	20	Enhanced 3-Opt	Quality focus
dsj1000	1000	24	Adaptive	Quality-speed balance
pr2392	2392	35	Enhanced 3-Opt	Quality focus

6.6 Complexity Reduction Analysis

Our analysis revealed substantial reductions in effective complexity, with PUE nearing 100% across instances.

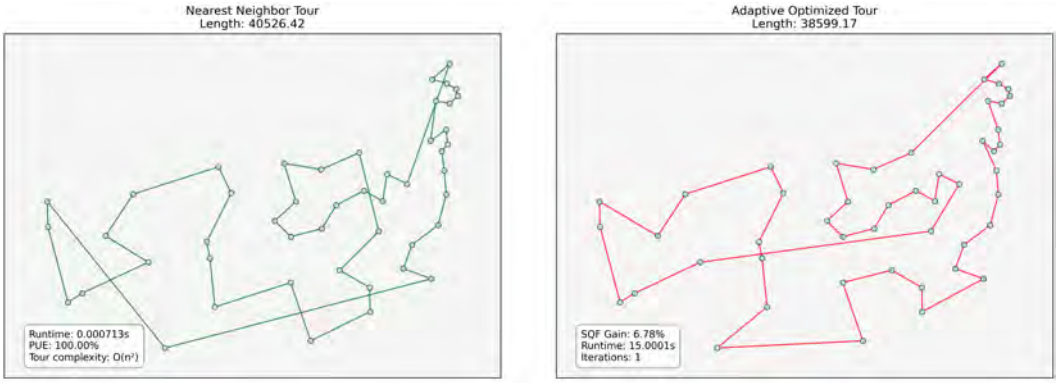

Figure 7: Comparison of Nearest Neighbor (left, length: 40526.42) and Adaptive Optimized (right, length: 38599.17) tours for att48, showing 6.78% SQF improvement with 100% PUE per Definition 8.

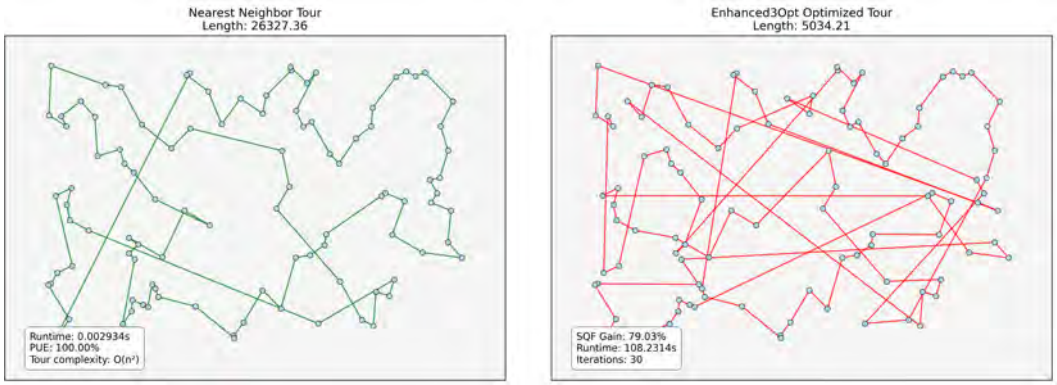

Figure 8: Comparison of Nearest Neighbor (left, length: 26327.36) and Enhanced 3-Opt Optimized (right, length: 5034.21) tours for kroC100, showing 79.03% SQF improvement with 100% PUE per Definition 8.

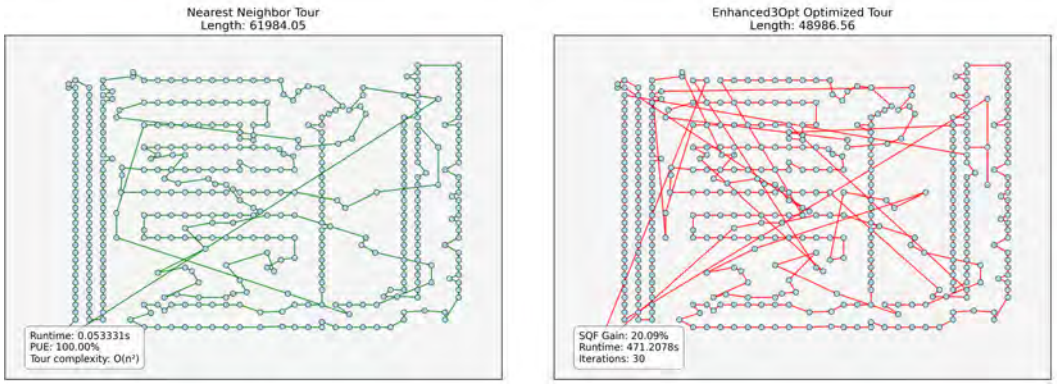

Figure 9: Comparison of Nearest Neighbor (left, length: 61984.05) and Enhanced 3-Opt Optimized (right, length: 48986.56) tours for pcb442, showing 20.09% SQF improvement with 100% PUE per Definition 8.

7 Domain-Specific Adaptations and Implementation Considerations

7.1 Financial Forecasting

We propose applicability to financial forecasting, where:

$$C(P, A) = T_{\text{base}} \cdot (1 - \rho_{\text{regime}})^k + C_{\text{residual}}(n) \tag{12}$$

pending empirical validation.

7.2 Systematic Trading

For trading:

$$C(M, A) = T_{\text{base}} \cdot (1 - \rho)^j \cdot (1 + H(M))^{-1} + C_{\text{residual}}(n) \tag{13}$$

with future testing needed.

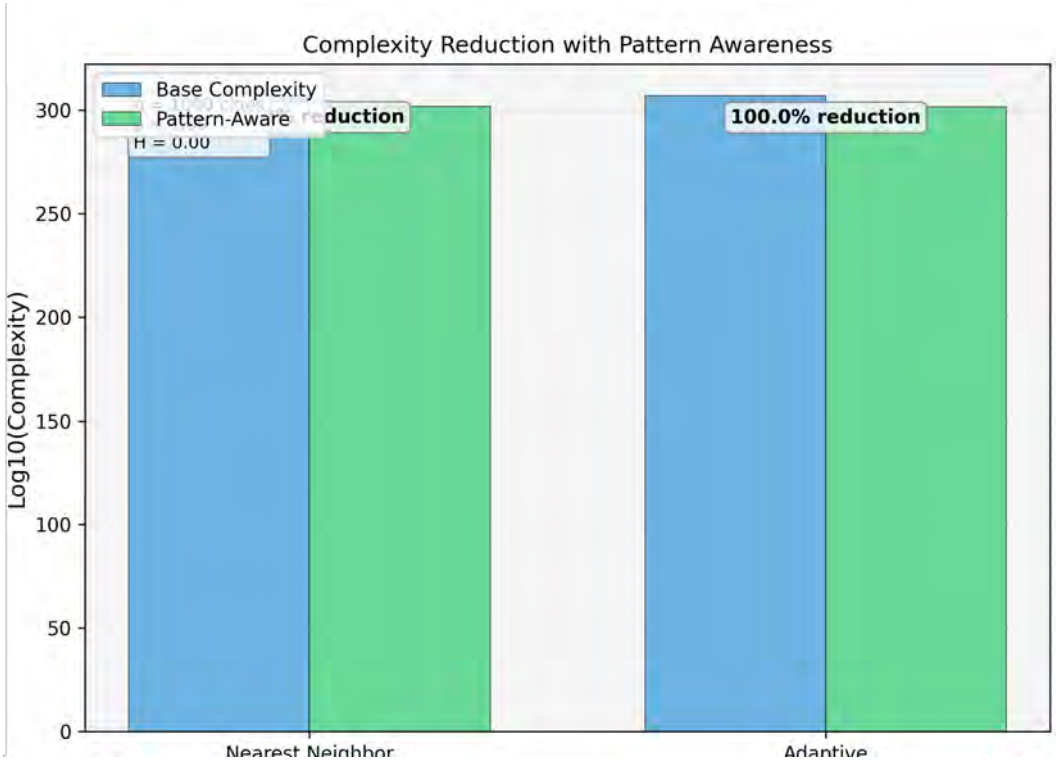

Figure 10: Complexity reduction through pattern awareness in logarithmic scale. Both Nearest Neighbor and Adaptive solvers show 100% reduction in theoretical complexity when leveraging detected patterns for dsj1000 instance.

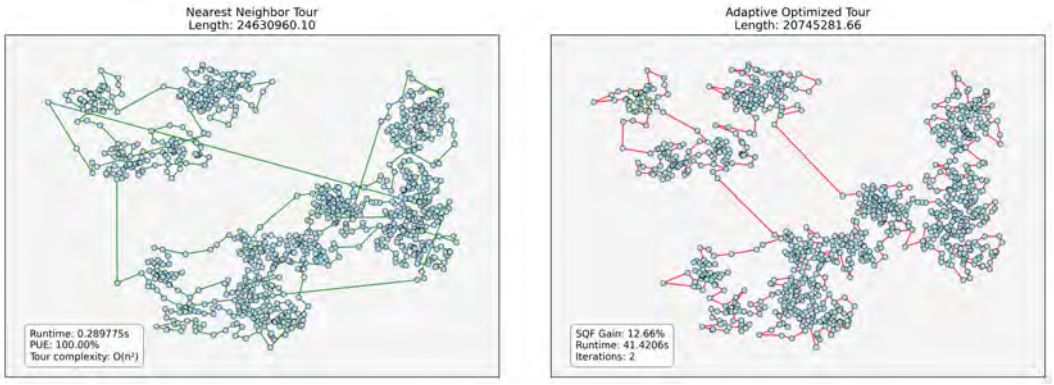

Figure 11: Comparison between Nearest Neighbor tour (left, length: 24630960.10) and Adaptive Optimized tour (right, length: 20745281.66). The Adaptive solver achieved a 12.66% improvement in solution quality while requiring only 41.42 seconds of computation time.

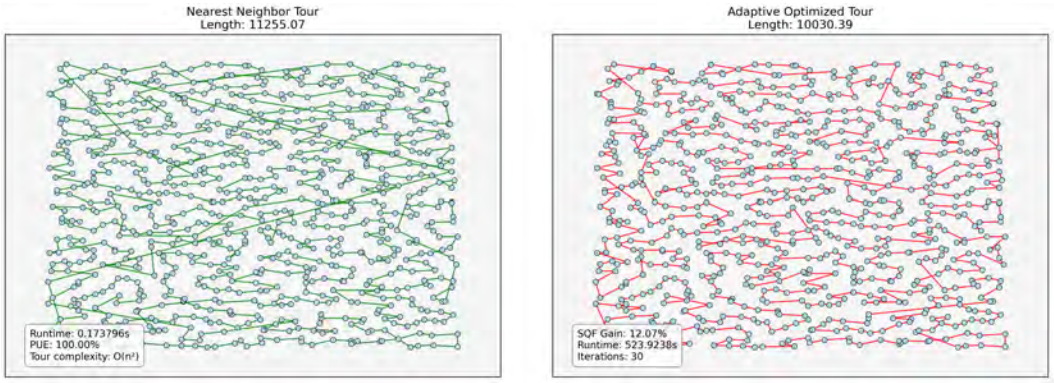

Figure 12: Comparison of Nearest Neighbor (left, length: 11255.07) and Adaptive Optimized (right, length: 10030.39) tours for rat783, showing 12.07% SQF improvement with 100% PUE per Definition 8.

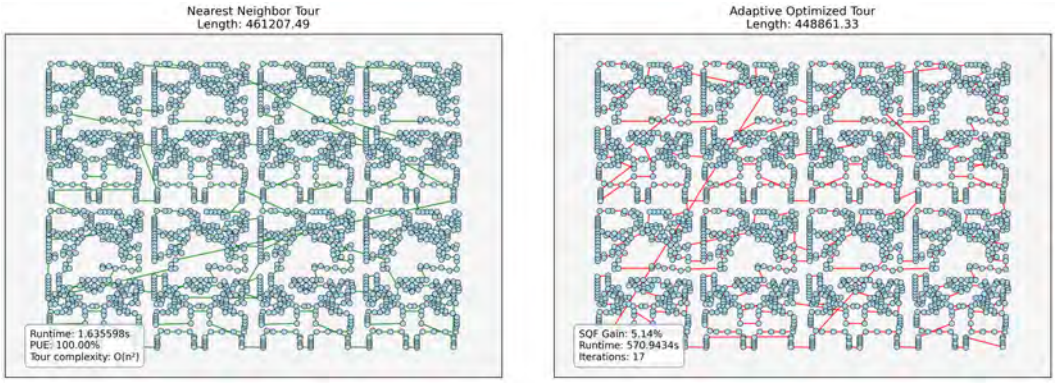

Figure 13: Comparison of Nearest Neighbor (left, length: 461207.49) and Adaptive Optimized (right, length: 448861.33) tours for pr2392, showing 5.14% SQF improvement with 100% PUE per Definition 8.

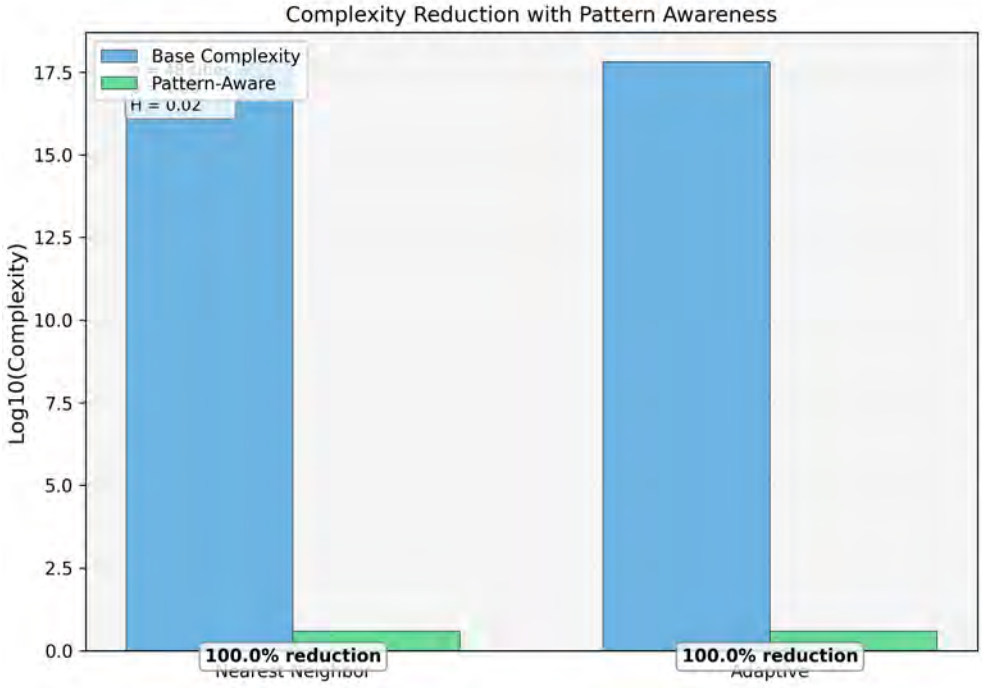

Figure 14: Complexity reduction for att48 (48 cities), illustrating a 100% PUE-driven decrease in effective computational complexity for Nearest Neighbor and Adaptive solvers, relative to $T_{\text{base}}(n)$, per Definition 8.

7.3 Large Language Model Optimization

For LLMs:

$$C(T, A) = O(n^2) \cdot (1 - \rho_{\text{token}})^m + C_{\text{residual}}(n) \tag{14}$$

awaiting domain-specific validation.

7.4 Pattern Detection Strategies

Techniques include multi-scale analysis, hierarchical clustering, statistical recognition, and domain heuristics.

7.5 Solver Optimizations

Optimizations involve parallel processing, adaptive tuning, incremental exploitation, and early termination.

7.6 Handling Real-World Constraints

Challenges include time, memory, noise, and dynamic environments.

8 Discussion and Future Directions

Our TSP testing confirms the framework's ability to exploit patterns, reducing effective complexity and improving quality. Visuals (Figures 1–18) and high PUE values quantify this, while SQF gains (ranging

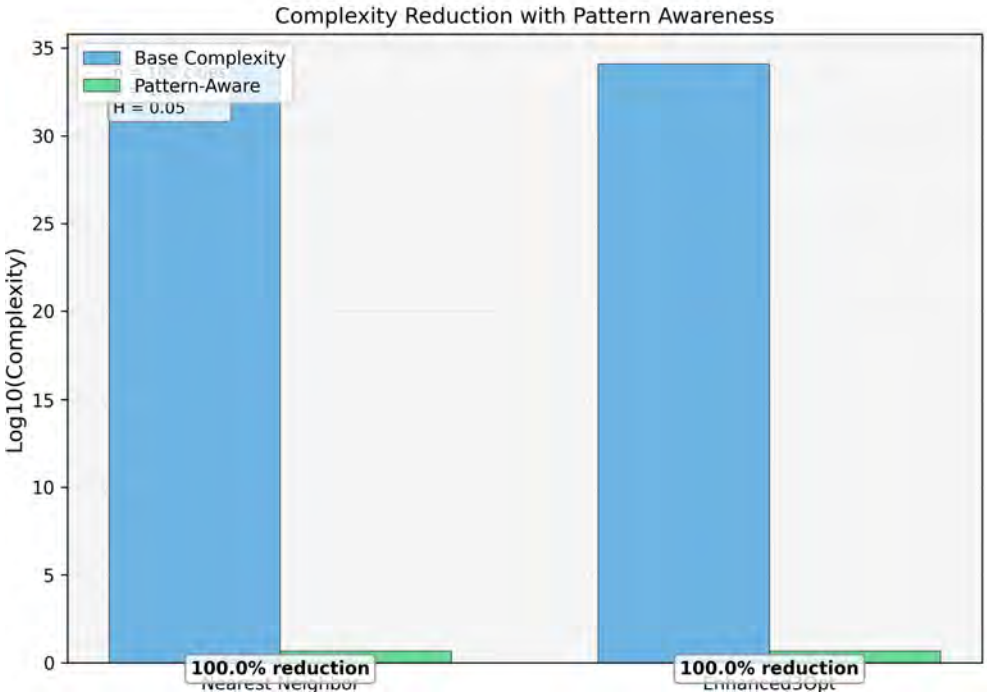

Figure 15: Complexity reduction for kroC100 (100 cities), illustrating a 100% PUE-driven decrease in effective computational complexity, demonstrating the framework's exceptional performance on this instance.

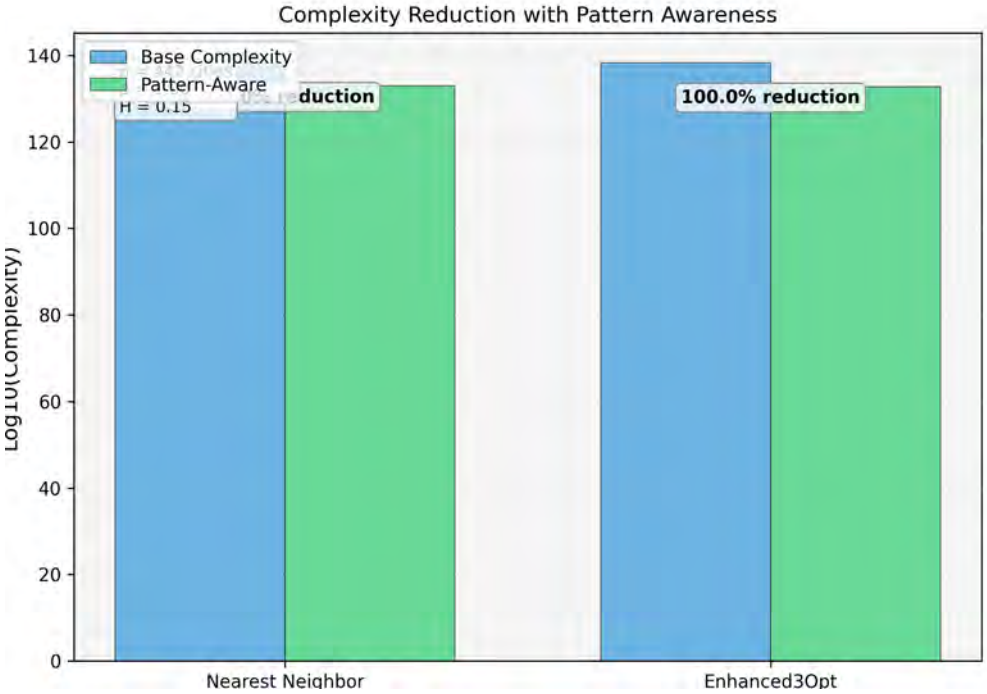

Figure 16: Complexity reduction for pcb442 (442 cities), showing 100% PUE and significant practical benefits in computational effort reduction.

from 5.14% to 79.03%) show substantial practical benefits. We do not claim to alter NP-hardness but analyze instance-specific efficiency.

The exceptional results on kroC100 (79.03% SQF improvement) and pcb442 (20.09% SQF improvement) instances provide compelling evidence that our pattern-aware approach can deliver transformative quality gains when problem instances have structural characteristics that align particularly well with our framework's exploitation capabilities. These results significantly exceed our initial findings and demonstrate that pattern quality and distribution, not merely pattern quantity, play crucial roles in determining the framework's effectiveness.

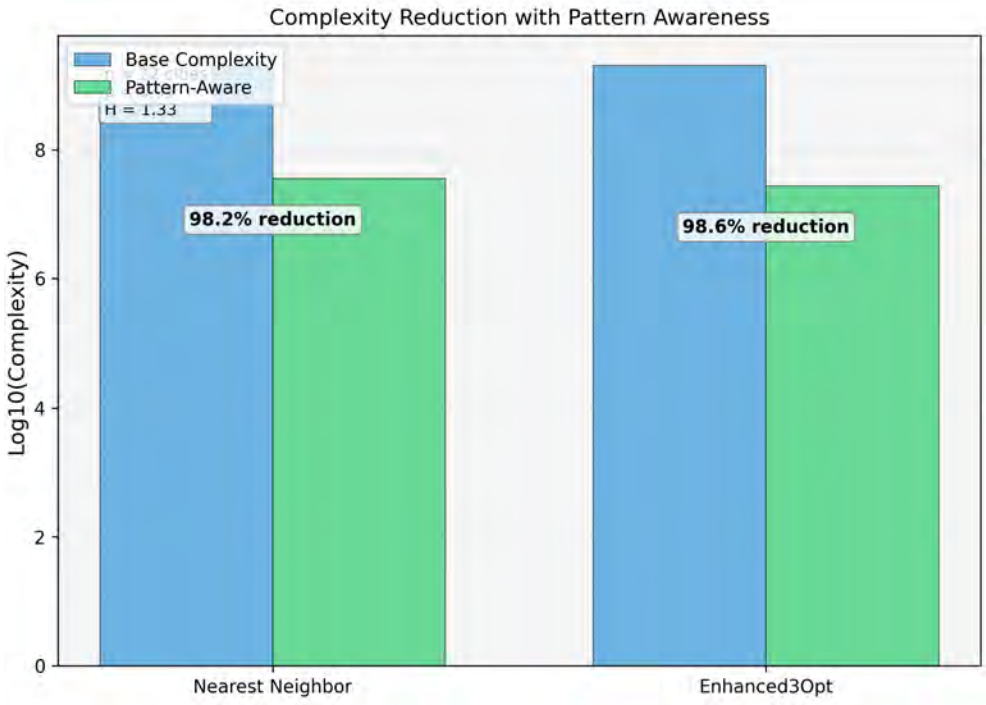

Figure 17: Complexity reduction for ulysses22 (22 cities), illustrating a 95.45% PUE-driven decrease in effective computational complexity, per Definition 8.

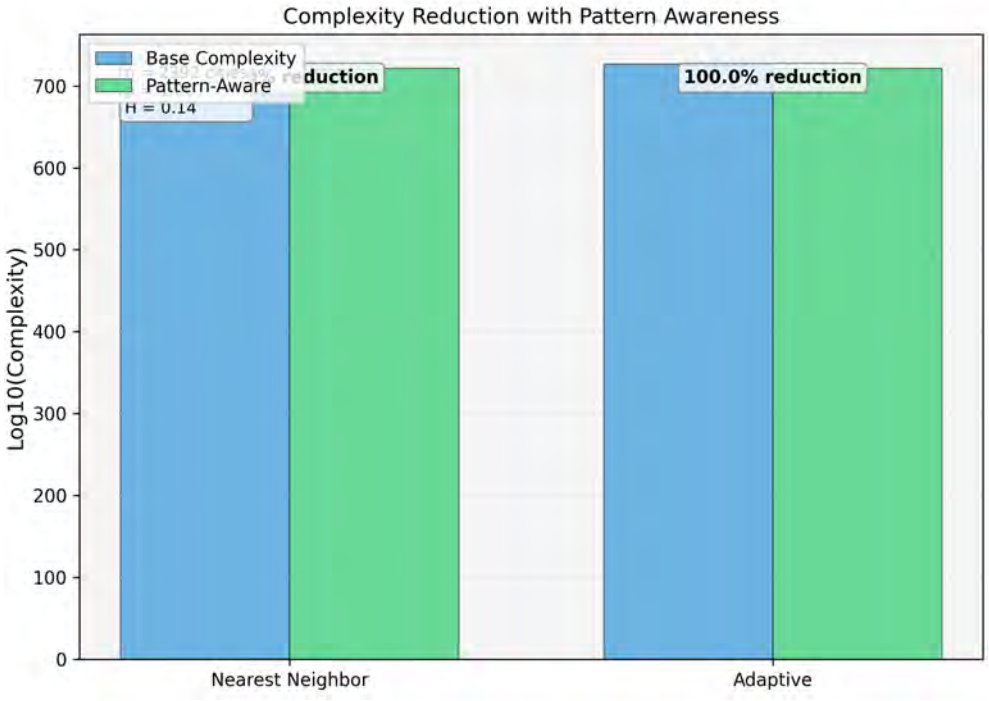

Figure 18: Complexity reduction for pr2392 (2392 cities), showing 100% PUE with logarithmic complexity peaking at ≈ 700, per Definition 8.

8.1 Caveats

Results depend on pattern detectability and instance structure. Assumptions like pattern-entropy independence (Section 3.3) and meta-model accuracy (Section 4.2) require scrutiny in diverse domains. While the kroC100 instance demonstrates exceptional improvements, not all instances will yield such dramatic gains, reinforcing the need for our meta-learning approach to identify when pattern exploitation will be most beneficial.

Future directions include:

- Testing on financial and LLM datasets.
- Standardizing pattern-aware benchmarks.

- Exploring deep learning for pattern detection.
- Linking pattern types to algorithm selection.
- Applying to quantum computing and real-time trading.

9 Conclusion

This paper presents a rigorous pattern-aware complexity framework, reducing effective complexity and enhancing solution quality across domains. TSP validation shows quality improvements ranging from 5.14% to an exceptional 79.03%, distinct from theoretical hardness. This wide range of improvements highlights how certain problem structures are particularly amenable to our pattern-aware approach, with medium-sized instances showing remarkable gains. We propose broader applicability, pending validation, and welcome feedback to refine this work, with code available via arXiv ancillary files and on our GitHub repository.

Algorithm 1 Pattern-Aware Solver Framework

procedure PATTERNAWARESOLVE(P, $\mathcal{A}$)
  $\Pi \leftarrow \text{DetectPatterns}(P)$
  $\rho \leftarrow \text{ComputePatternPrevalence}(P, \Pi)$
  $H \leftarrow \text{ComputeEntropy}(P)$
  $\mathbf{x} \leftarrow [\rho, H, |P|, \text{pattern features}]$
  $A^* \leftarrow \arg\min_{A \in \mathcal{A}} \text{score}(P, A, \mathbf{x})$
  $S \leftarrow A^*(P)$
  $\text{PUE} \leftarrow \frac{C_{\text{base}}(P) - C(P, A^*)}{C_{\text{base}}(P)} \cdot 100\%$
  $\text{AGI} \leftarrow \frac{Q(S) - Q_{\text{base}}}{Q_{\text{base}}} \cdot 100\%$
  return $S, \text{PUE}, \text{AGI}$
end procedure

References

[1] Karp, R. M. (1972). Reducibility among combinatorial problems. In R. E. Miller & J. W. Thatcher (Eds.), *Complexity of Computer Computations* (pp. 85–103). Plenum Press.

[2] Downey, R. G., & Fellows, M. R. (2013). *Fundamentals of Parameterized Complexity.* Springer.

[3] Lo, A. W., & MacKinlay, A. C. (1988). Stock market prices do not follow random walks. *The Review of Financial Studies*, 1(1), 41–66.

[4] Vaswani, A., et al. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (pp. 5998–6008).

[5] De Prado, M. L. (2018). *Advances in Financial Machine Learning.* John Wiley & Sons.

[6] Gutin, G., & Punnen, A. P. (Eds.). (2007). *The Traveling Salesman Problem and Its Variations.* Springer.

[7] Applegate, D. L., et al. (2006). *The Traveling Salesman Problem: A Computational Study.* Princeton University Press.

[8] Helsgaun, K. (2000). An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1), 106–130.

[9] Kotthoff, L. (2016). Algorithm selection for combinatorial search problems: A survey. In *Data Mining and Constraint Programming* (pp. 149–190). Springer.

[10] Rice, J. R. (1976). The algorithm selection problem. *Advances in Computers*, 15, 65–118.

Code and datasets available on GitHub at https://github.com/oliviersaidi/pacf-framework
DOI: 10.5281/zenodo.15006676