

Supplementary Material

Predicting Effective Heuristics and Optimality Gap for the Traveling Salesman Problem Using Lightweight Instance Features

This document collects supporting diagnostics, reference tables, and raw measurements for the main manuscript. None of it is required to follow the paper’s results: every finding reported here is stated in the main text, and this material supplies the underlying numbers for readers who wish to audit them. Each section names the passage of the main text it supports.

Accompanying data files. Four CSVs accompany this document.

`supplementary_stage1_oof_predictions.csv`

All 160 instances with their out-of-fold Stage 1 predictions from all three selectors: each selector’s predicted heuristic, the cost of that heuristic, and the resulting regret. Regret is left empty exactly where it is undefined, which is what produces the 160/159/159 denominators used throughout the Results.

`supplementary_stage2_predictions.csv`

All 32 Seed-1 test instances with a complete column block per pipeline: predicted heuristic, realized gap of the selected heuristic, point prediction, interval bounds, interval width, error, and coverage flag. Each pipeline carries its *own* realized-gap column, because that quantity is the gap of the heuristic that pipeline’s Stage 1 selected, and therefore differs between the two backbones on the three instances where they disagree. Every MAE, R^2 and coverage figure in Table 5 of the main text can be recomputed from this file.

`timeout_instances.csv`

The 23 excluded instance–heuristic pairs, also tabulated in Table S10.

`hyperparameter_sensitivity.csv`

The full hyperparameter sweep, also tabulated in Tables S7 and S8.

S1. Abbreviations

S2. Stage 1 per-class performance and confusion matrix

Supports *Results* — *Random Forest*, which reports per-class precision, recall and F1, and the finding that Nearest Neighbor is never recovered out of fold. All figures come from the aggregated out-of-fold predictions over all 160 instances (Protocol A).

Table S1 | Abbreviations used in the manuscript and in this document. Each is also expanded at its first use in the main text.

Abbreviation	Expansion
TSP	Traveling Salesman Problem
NP	Nondeterministic Polynomial time
TSPLIB	Reinelt’s library of benchmark TSP instances
NN	Nearest Neighbor (heuristic, and Stage 1 class label)
CH	Christofides (heuristic, and Stage 1 class label)
MST	Minimum Spanning Tree
LKH	Lin–Kernighan–Helsgaun
RF	Random Forest
XGB	XGBoost
LR	Logistic Regression
CV	Cross-validation. Never coefficient of variation, which is written out
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
R^2	Coefficient of determination
$R^2(\log)$	Coefficient of determination computed on $\log(1 + g)$
QR	Quantile Regression
pp	Percentage points
VIF	Variance Inflation Factor
d_z	Cohen’s d for paired samples
S	Monte Carlo sample size for the triangle-inequality features (200,000 triples)

Table S2 | Random Forest per-class performance from aggregated out-of-fold predictions over all 160 instances. Nearest Neighbor, with only 8 positive examples, is never predicted at all, so its precision is undefined in the usual sense and is reported as 0.00 by convention.

Class	Support	Precision	Recall	F1
Christofides	75	0.81	0.91	0.86
Insertion	46	0.80	0.87	0.83
Greedy	31	0.77	0.65	0.70
Nearest Neighbor	8	0.00	0.00	0.00

Table S3 | Random Forest confusion matrix, aggregated out-of-fold predictions over all 160 instances. The empty Nearest Neighbor column is the structural consequence of that class holding only 8 of 160 instances: no fold contains enough positive examples for the classifier ever to predict it. The largest off-diagonal cell is nine true-Greedy instances routed to Christofides, which is also among the least costly error modes, as Table S4 shows.

True	Predicted				Total
	CH	Insertion	Greedy	NN	
Christofides	68	3	4	0	75
Insertion	5	40	1	0	46
Greedy	9	2	20	0	31
Nearest Neighbor	2	5	1	0	8
Total	84	50	26	0	160

S3. Stage 1 case studies

Supports *Results — Random Forest*, which reports that most misclassifications fall between heuristics of comparable cost. Correctly and incorrectly classified instances appear side by side with the predicted heuristic, the true best heuristic, and the resulting regret. Full detail for all 160 instances is in `supplementary_stage1_oof_predictions.csv`.

Table S4 | Representative Stage 1 case studies from the out-of-fold predictions. Regret is the relative excess tour cost of the predicted heuristic over the best feasible one, so a correct prediction has regret exactly zero. The contrast between the two selectors is the point: outside the single adversarial `christofides_hard` instance, Random Forest’s worst error costs 0.149, whereas Logistic Regression’s worst costs 14.632 — on `brg180` it selects Greedy at a cost of 37,830 against Insertion’s 2,420.

Instance	Type	<i>n</i>	Selector	True best	Predicted	Regret
<i>Correctly classified</i>						
<code>a280</code>	EUC_2D	280	RF	Christofides	Christofides	0.000
<code>bays29</code>	Explicit	29	RF	Insertion	Insertion	0.000
<code>family_A_random_n100_s0</code>	Explicit	100	RF	Greedy	Greedy	0.000
<i>Misclassified — Random Forest, ranked by regret</i>						
<code>christofides_hard_n50_s0</code>	Explicit	50	RF	Christofides	Insertion	3.168
<code>eil51</code>	EUC_2D	51	RF	Christofides	Greedy	0.149
<code>pr144</code>	EUC_2D	144	RF	Nearest Nb.	Christofides	0.145
<code>gr48</code>	Explicit	48	RF	Insertion	Greedy	0.145
<code>fl1400</code>	EUC_2D	1400	RF	Christofides	Greedy	0.082
<i>Misclassified — Logistic Regression, ranked by regret</i>						
<code>brg180</code>	Explicit	180	LR	Insertion	Greedy	14.632
<code>christofides_hard_n50_s0</code>	Explicit	50	LR	Christofides	Insertion	3.168
<code>bays29</code>	Explicit	29	LR	Insertion	Greedy	0.155

S4. Feature correlation matrix

Supports *Results — Feature Diagnostics and Collinearity*, which reports that three feature pairs exceed $|r| = 0.7$ and that variance inflation factors reach 84.12. Per-feature importances and VIFs are in Table 4 of the main text; the full pairwise matrix behind them is below.

Read alongside Table 4 of the main text, the two triangle-inequality features carry nearly the same information, so the apparent gap between their Gini scores (0.185 against 0.175) is not meaningful; they are better read as jointly carrying roughly 0.36 of Gini importance. The same redundancy explains why permutation importance ranks `n` first while Gini ranks the triangle features first: permuting one of two correlated features leaves the model able to recover the signal from the other, understating both, whereas `n` is nearly uncorrelated with everything else (maximum $|r| = 0.319$) and suffers no such effect. The disagreement diagnoses redundancy rather than unreliability in either measure.

Table S5 | Pearson correlation matrix over the eight numeric features, computed across all 160 instances after the normalization described in the main text. Column abbreviations match the row labels. Three pairs exceed $|r| = 0.7$: triangle_violation_rate–avg_violation_magnitude (0.914), cv_edge_weight–triangle_violation_rate (0.846), and min_edge_weight–mst_weight (0.780). The categorical type indicator is excluded, as Pearson correlation is defined for the numeric features only.

	n	min	cv	skew	pct	tri	avg	mst
n	1.000	−0.155	−0.107	0.098	−0.077	−0.183	−0.227	−0.319
min_edge_weight	−0.155	1.000	−0.322	0.072	−0.006	−0.220	−0.201	0.780
cv_edge_weight	−0.107	−0.322	1.000	0.028	0.115	0.846	0.645	−0.470
edge_weight_skewness	0.098	0.072	0.028	1.000	−0.065	−0.384	−0.486	0.310
pct_short_edges	−0.077	−0.006	0.115	−0.065	1.000	0.127	0.373	−0.048
triangle_violation_rate	−0.183	−0.220	0.846	−0.384	0.127	1.000	0.914	−0.425
avg_violation_magnitude	−0.227	−0.201	0.645	−0.486	0.373	0.914	1.000	−0.355
mst_weight	−0.319	0.780	−0.470	0.310	−0.048	−0.425	−0.355	1.000

S5. Provenance signal carried by individual features

Supports *Results* — *XGBoost*, which argues that feature importance cannot settle whether the classifier depends on instance provenance, because several numeric features redundantly encode it. Each row fits a depth-2 decision tree predicting whether an instance is Explicit from that one feature alone, under the same stratified 5-fold protocol used throughout (`shuffle=True`, `random_state=1`).

Table S6 | Provenance is recoverable from any one of three numeric features on its own, which is why Gini credit shifts arbitrarily among them, and why the direct type ablation — not feature importance — is what settles the provenance-shortcut question in the main text. Reproduced by `experiments/provenance_single_feature.py`.

Single feature	CV accuracy predicting is Explicit
Majority baseline	55.6%
<code>cv_edge_weight</code>	81.9%
<code>min_edge_weight</code>	85.0%
<code>avg_violation_magnitude</code>	94.4%

S6. Hyperparameter sensitivity sweep

Supports *Results* — *XGBoost*, which reports that only one of the twenty XGBoost cells beats the adopted configuration, by a margin well inside fold-to-fold noise.

Each cell gives mean cross-validation accuracy $\pm$ standard deviation across the five stratified folds over all 160 instances, with every other hyperparameter held at its published value. The adopted configuration is marked †. Cell values are rounded to two decimals for display; differences quoted in the text are computed from the unrounded fold means.

Table S7 | XGBoost hyperparameter sweep, cross-validation accuracy (%). The grid spans 5.01 percentage points end to end. Only one cell (depth 2, 50 rounds) exceeds the adopted configuration, by 1.25 points (80.625% against 79.375% unrounded), against a mean within-grid fold-to-fold standard deviation of 4.54 points — so the margin is under a third of a standard deviation and cannot be separated from resampling noise at 160 instances.

XGBoost max depth	Boosting rounds			
	50	100	200	500
2	80.63 $\pm$ 3.64	79.37 $\pm$ 5.08	78.75 $\pm$ 5.38	76.88 $\pm$ 4.24
3	78.75 $\pm$ 6.37	79.37 $\pm$ 6.12 †	77.50 $\pm$ 4.15	75.62 $\pm$ 3.64
4	78.12 $\pm$ 5.23	78.12 $\pm$ 5.23	76.88 $\pm$ 3.19	75.62 $\pm$ 3.64
5	78.75 $\pm$ 5.00	78.12 $\pm$ 5.23	76.88 $\pm$ 3.19	75.62 $\pm$ 3.64
6	78.12 $\pm$ 5.23	78.12 $\pm$ 5.23	76.88 $\pm$ 3.19	76.25 $\pm$ 4.24

Table S8 | Random Forest hyperparameter sweep, cross-validation accuracy (%). No cell exceeds the adopted configuration, and the grid spans only 1.25 percentage points end to end against XGBoost’s 5.01, corroborating that Random Forest reaches competitive accuracy without a tuning search. Mean within-grid fold-to-fold standard deviation is 5.14 points.

Random Forest max depth	Number of trees			
	100	200	400	800
None	80.00 $\pm$ 5.08	80.00 $\pm$ 5.08	80.00 $\pm$ 5.08 [†]	80.00 $\pm$ 5.08
5	78.75 $\pm$ 6.06	80.00 $\pm$ 5.08	79.37 $\pm$ 5.08	80.00 $\pm$ 5.08
10	80.00 $\pm$ 5.08	80.00 $\pm$ 5.08	80.00 $\pm$ 5.08	80.00 $\pm$ 5.08
20	80.00 $\pm$ 5.08	80.00 $\pm$ 5.08	80.00 $\pm$ 5.08	80.00 $\pm$ 5.08

S7. Prediction-interval widths and calibration

Supports *Results — Statistical Significance*, which reports that XGBoost + Quantile is significantly miscalibrated (Holm $p = 0.0008$) while neither conformal pipeline deviates significantly from nominal ($p = 0.4578$ and $p = 0.9210$).

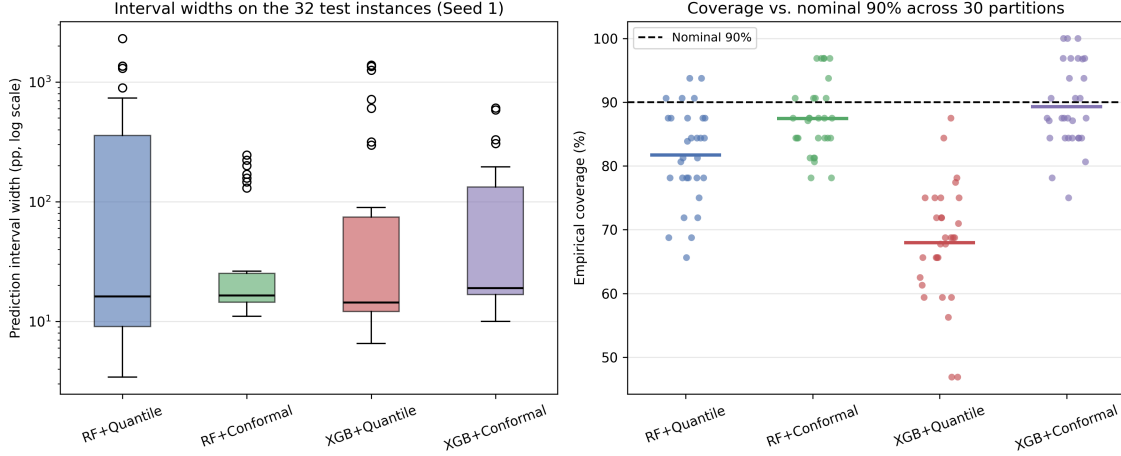


Figure S1 | Left: distribution of prediction-interval widths across the 32 Seed 1 test instances, on a logarithmic scale. The conformal pipelines produce tightly concentrated widths, a direct consequence of applying a single instance-independent correction, whereas the quantile pipelines adapt width to the instance and produce extreme outliers exceeding 1,000pp. Right: empirical coverage against the nominal 90% target, one point per seed across the thirty repeated hold-out partitions, horizontal bars marking across-seed means. The XGBoost quantile pipeline falls below the nominal line on nearly every partition, the visual form of its significant miscalibration (Holm $p = 0.0008$); both conformal pipelines straddle the line and are statistically indistinguishable from nominal ($p = 0.4578$ and $p = 0.9210$).

S8. Timing measurements by instance size

Supports *Results — Computational Cost of Feature Extraction*, which reports a mean speedup of $15.86\times$, a net loss below roughly $n = 150$, and $62.73\times$ on the largest instances measured.

All times are per-instance wall-clock measurements taken on a single machine with a monotonic timer. Of the 155 instances covered by the timing benchmark, the 130 with $n \leq 800$ had all four heuristics run to completion and form the measured set.

S9. Timed-out instance–heuristic pairs

Supports *Results — Computational Cost of Feature Extraction*, closing paragraph. Two thresholds are easy to conflate: the 600,000ms cap governs Stage 1’s best-heuristic labelling, and 23 pairs exceed it; the 1,000,000ms sentinel marks runs that did not finish, leaving the optimality gap undefined. Sixteen of the 23 carry the sentinel exactly; the other 7 have measured runtimes between cap and sentinel. The Stage 2 gap table therefore contains 624 rows rather than $640 - 23 = 617$.

Table S9 | Feature extraction plus two-stage inference against running all four heuristics, by instance size. Feature-extraction time is nearly flat in n because the two Monte Carlo triangle-inequality features draw a fixed 200,000 triples regardless of instance size and account for 98.2% of extraction time; heuristic time grows steeply, which is what produces the monotone trend in speedup. The framework is a net loss below roughly $n = 150$ and improves sharply above it. Speedup is the mean of the per-instance ratios, not the ratio of the medians shown.

Size bucket	Instances	Median feature extraction (ms)	Median all four heuristics (ms)	Mean speedup
$n \leq 130$	45	391.1	39.3	$0.52\times$
$131 \leq n \leq 300$	44	393.3	761.3	$5.56\times$
$301 \leq n \leq 500$	20	439.1	13,078.2	$23.84\times$
$501 \leq n \leq 800$	21	482.9	28,100.4	$62.73\times$
All	130	—	—	$15.86\times$

Table S10 | All 23 instance–heuristic pairs exceeding the 600,000ms cap. All are Insertion (15) or Christofides (8); Nearest Neighbor and Greedy never time out. Every affected instance has $n \geq 1577$, and below $n = 2319$ only Insertion exceeds the cap, consistent with both heuristics’ $O(n^3)$ cost. Also supplied as `timeout_instances.csv`.

Instance	n	Heuristic	Runtime (ms)	Sentinel
f11577	1577	Insertion	787,284	
d1655	1655	Insertion	910,736	
vm1748	1748	Insertion	1,102,586	
u1817	1817	Insertion	1,186,843	
r11889	1889	Insertion	1,370,244	
d2103	2103	Insertion	1,000,000	✓
u2152	2152	Insertion	1,000,000	✓
u2319	2319	Christofides	733,194	
u2319	2319	Insertion	1,000,000	✓
pr2392	2392	Christofides	667,073	
pr2392	2392	Insertion	1,000,000	✓
pcb3038	3038	Christofides	1,000,000	✓
pcb3038	3038	Insertion	1,000,000	✓
nu3496	3496	Christofides	1,000,000	✓
nu3496	3496	Insertion	1,000,000	✓
f13795	3795	Christofides	1,000,000	✓
f13795	3795	Insertion	1,000,000	✓
fn14461	4461	Christofides	1,000,000	✓
fn14461	4461	Insertion	1,000,000	✓
r15915	5915	Christofides	1,000,000	✓
r15915	5915	Insertion	1,000,000	✓
r15934	5934	Christofides	1,000,000	✓
r15934	5934	Insertion	1,000,000	✓

S10. Scale-invariance verification

Supports *Results — Validation and Robustness*, “Scale invariance”. 154 of the 160 dataset instances (the remaining six lack locally available source files: bayg29, qa194, nu3496, lu980, zi929, five_d) were rescaled by 0.01, 10 and 1000; the full feature vector was recomputed from the rescaled matrix and passed through both trained stages. All 308 tests at $10\times$ and $1000\times$ reproduce the unscaled prediction exactly and are omitted below; only the 67 mismatches, all at $0.01\times$, are shown.

Table S11 | Scale-invariance mismatches under $0.01\times$ rescaling, out of 154 tested instances.

The four rows shown are the only cases where the discrete heuristic decision itself changes; the remaining 63 mismatches leave the heuristic decision unchanged but the predicted gap differs by the amounts summarized above. Zero mismatches of any kind occur at $10\times$ or $1000\times$, across 308 tests. All are traced to one intoleranced floating-point comparison in the triangle-inequality check: on near-exactly-metric instances, sampled triples within rounding error of equality can flip across the strict inequality once every distance is shrunk by two orders of magnitude. A tolerance-based comparison would remove it. Reproduced by `experiments/verify_scale_invariance_full.py`; full per-instance results in `scale_invariance_predictions_full.csv`.

Instance	n	Type	Base heuristic	Scaled heuristic	Rel. gap diff.
burma14	14	Geo	Insertion	CH	0.00%
u1432	1432	EUC_2D	Insertion	CH	6.60%
eil101	101	EUC_2D	Insertion	CH	11.96%
rat195	195	EUC_2D	Greedy	CH	13.39%
(63 further instances: heuristic decision unchanged, gap differs by 2.65% relative on median, 16.40% at most)					

S11. Leave-one-family-out generalization

Supports *Results — Validation and Robustness*, “Generalization across instance families”. Each row withholds one complete family from training and evaluates on it, against that family’s own majority-class baseline.

Table S12 | Leave-one-family-out results. The classifier never exceeds the held-out family’s majority baseline, and on the two families whose composition differs most from the training remainder it falls far below. This is the strictest generalization test in the paper and the framework does not pass it; the claim in the main text is bounded accordingly. Table S13 shows the cause is absent training coverage rather than a collapsing shortcut.

Held-out family	n	Accuracy	Majority baseline	Verdict
Random (generated)	13	0.00%	100.00%	far below baseline
Clustered (generated)	13	76.92%	76.92%	exactly at baseline
christofides_hard	29	86.21%	86.21%	exactly at baseline
TSPLIB	105	17.14%	70.48%	far below baseline

Table S13 | Model-agnostic 1-nearest-neighbor control in standardized feature space. For each held-out instance we take its single nearest training instance and ask whether the label matches. A procedure that performs no learning at all fails at rates comparable to the classifier, and the nearest training instance sits 6 to 9 times farther away than a typical within-dataset neighbor gap (0.307). The failure is therefore absent local training coverage, not a spurious correlate collapsing.

Held-out family	n	1-NN label match rate	Mean 1-NN distance	Ratio to typical neighbor gap
Random (generated)	13	7.7%	1.830	$6.0\times$
Clustered (generated)	13	7.7%	2.107	$6.9\times$
christofides_hard	29	10.3%	2.828	$9.2\times$
TSPLIB	105	16.2%	2.813	$9.2\times$

S12. Stage 1 to Stage 2 error propagation

Supports *Results — Validation and Robustness*, “Stage 1 to Stage 2 error propagation”. Each pipeline was run twice on the identical Seed-1 split: as deployed, receiving Stage 1’s predicted heuristic, and as an oracle receiving the true best heuristic.

Table S14 | Deployed versus oracle Stage 2 performance. Negative ΔMAE means the deployed pipeline is more accurate than one given perfect Stage 1 labels, which holds for three of the four pipelines. The explanation is what each configuration is scored against: the deployed pipeline against the true gap of the heuristic it selected, the oracle against that of the genuinely best heuristic. Random Forest’s Seed-1 errors are four true-Greedy instances predicted as Christofides, so the oracle must predict Greedy gaps on them — the scarcer and harder-to-model class, at 31 training instances against Christofides’ 75 — while the deployed pipeline predicts the better-modeled Christofides gaps. XGBoost + Quantile’s misclassifications include cases for which no such favorable substitution holds. End-to-end gap estimates are therefore not systematically degraded by imperfect selection, but this reflects which instances are misclassified and how well their substituted heuristics are modeled, not a general guarantee.

Pipeline	Deployed MAE (pp)	Oracle MAE (pp)	ΔMAE (pp)	Deployed $R^2(\log)$	Oracle $R^2(\log)$
RF + Quantile	5.21	6.29	−1.08	0.9204	0.8200
RF + Conformal	16.43	17.08	−0.65	0.7948	0.7240
XGB + Quantile	17.95	12.07	+5.88	0.8580	0.7396
XGB + Conformal	47.73	49.14	−1.42	−0.0522	−0.4088