TSP Solver: Comparative Analysis of Exact, Approximate, and Heuristic Methods

Yixuan Li, Yitong Wu, NaiJen Cheng

1 Algorithm Summaries

1.1 Brute Force (BF) – Exact

The brute-force solver enumerates all Hamiltonian tours to guarantee the optimal solution. We fix the first city as the starting point and generate all permutations of the remaining $n-1$ cities. For each permutation, we compute the total tour cost and update the best-so-far solution. Because the algorithm explores $O(n!)$ routes, a time cutoff is necessary for all but the smallest instances.

Pseudo-code (simplified).

```
best = INF
for perm in permutations(cities[1..n]):
    cost = route_cost(perm)
    if cost < best:
        best = cost
    if time > cutoff: break
return best
```

Complexity: $O(n!)$.
Full tour? Only if the search completes before the cutoff.

1.2 MST-Based 2-Approximation (Approx) – Deterministic

This algorithm constructs a Minimum Spanning Tree via Prim's algorithm, performs a DFS preorder traversal, and returns the visiting order as a tour. For metric TSP, the resulting route is provably within $2 \times OPT$.

Pseudo-code.

```
MST   = Prim(G)
order = DFS(MST)
return make_tour(order)
```

Complexity: $O(n^2)$.
Full tour? Always.

1.3 Simulated Annealing + 2-opt Local Search (LS) – Heuristic

We first build an initial solution via Nearest Neighbor, then repeatedly apply 2-opt moves. Worse solutions are accepted with probability $e^{-\Delta/T}$, and the temperature decays from 10000 to 0.1 with rate 0.9995 until termination.

Pseudo-code.

```
tour = NN_initial()
T    = 10000
while T > 0.1:
```

```
    (i, j) = pick_2opt()
     = cost(new) - cost(tour)
    if  < 0 or rand() < exp(-/T):
        tour = new
    T *= 0.9995
return tour
```

Complexity: Typically < 1 s.
Full tour? Always.
Averaging: LS results averaged over 10 seeds (0–9).

2 Performance Comparison

A complete results table is provided in `results.csv`. For LS, each entry reports the averaged value across 10 runs. The `RelError` column is computed relative to the best known solution (BF optimal for $n \leq 10$, LS best otherwise). We include the full table here for completeness; the following is the entire results table. These examples highlight the scalability gap between BF and LS.

Sample Results

- **Cincinnati (10 cities).** BF = 277,952 (optimal, 3.7 s), LS = 277,952 (0.3 s avg, 0% error).
- **Atlanta (20 cities).** BF = 3,353,390 (67% error), LS = 2,024,401 (1% error).
- **Roanoke (230 cities).** BF = 6,849,948 (763% error), LS = 823,456 (4% error).

Instance	Size	BF_Time(s)	BF_Qualit	BF_RelErr	Approx_Tin	Approx_Qua	Approx_Rel	LS_Time(s)	LS_Qualit	LS_RelErr	LS_Runs	LS_Min	LS_Max
Atlanta	20	300	3353390	67.35	<1	2380448	18.8	~1	2024401.4	1.03	10	2003763	2045745
Berlin	52	300	19249	146.43	<1	10402	33.17	~1	8175	4.66	10	7811	8439
Boston	40	300	2220896	147.15	<1	1150963	28.08	~1	913481.4	1.66	10	898594	955656
Champaign	55	300	209943	295.54	<1	65712	23.81	~1	54049.3	1.83	10	53077	56748
Cincinnati	10	300	277952	0	<1	301216	8.37	~1	277952	0	10	277952	277952
Denver	83	300	546699	419.49	<1	134748	28.04	~1	107625.5	2.27	10	105238	110081
NYC	68	300	7166313	355.19	<1	2027107	28.76	~1	1624851.6	3.21	10	1574363	1662583
Philadelph	30	300	3710782	165.82	<1	1646249	17.93	~1	1405273	0.67	10	1395981	1433649
Roanoke	230	300	6849948	763.16	<1	838282	5.63	~1	823456.5	3.76	10	793588	840996
SanFrancis	99	300	5697031	581.05	<1	1134989	35.68	~1	865935.8	3.52	10	836512	883497
Toronto	109	300	9219828	682.63	<1	1675105	42.19	~1	1223719.7	3.88	10	1178064	1267306
UKansasSta	10	300	62962	0	<1	68090	8.14	~1	62962	0	10	62962	62962
UMissouri	106	300	670811	369.62	<1	178249	24.79	~1	146290.8	2.41	10	142842	150817

Figure 1: Results

3 Effect of Cutoff Time on Solution Quality

3.1 Brute Force (BF): Extremely Sensitive

BF performance collapses under fixed cutoff constraints due to its factorial complexity:

- $n \leq 10$: optimal results within seconds.
- $n = 20$–40: 300 s explores $< 0.001\%$ of the search space; errors reach 67–147%.
- $n \geq 50$: cutoff prevents completing even a single full tour; errors exceed 400–700%.

Increasing cutoff time offers negligible improvement.

3.2 Local Search (LS): Cutoff-Independent

Simulated annealing converges based on temperature, not wall-clock time:

- Convergence occurs within 0.3–0.7 s for all instances.
- Extending cutoff to 60 or 600 s yields identical solutions.
- Variance across seeds remains low (1–4%).

Hence LS is effectively independent of cutoff.

3.3 Approximation Algorithm

The MST-based approximation always completes in under 1 s; cutoff time has no effect.

4 Key Findings and Implementation Details

- **Solution Quality:** LS $\gg$ Approx $>$ BF.
- **Runtime:** Approx $\approx$ LS $<$ 1 s $\ll$ BF.
- **Scalability:** LS and Approx handle 200+ cities; BF fails beyond 12–15.

Implementation. Python 3 implementation; main executable in `code/exec.py`. Local search results are averaged over random seeds 0–9. The file `results.csv` contains the full performance table required for submission.